NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

THE THRESHOLD SHORTEST PATH INTERDICTION PROBLEM FOR CRITICAL INFRASTRUCTURE RESILIENCE ANALYSIS

by

Charles R. Clark

September 2017

Thesis Advisor:	W. Matthew Carlyle
Second Reader:	David L. Alderson

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704–0188*

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE September 2017	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE THE THRESHOLD SHORTEST PATH INTERDICTION PROBLEM FOR CRITICAL INFRASTRUCTURE RESILIENCE ANALYSIS	5. FUNDING NUMBERS
6. AUTHOR(S) Charles R. Clark	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this document are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: N/A.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

We formulate and solve the threshold shortest path interdiction problem, which we define as follows: Find a finite set of arcs to attack within a network such that the resulting shortest path from a given source node to a given destination is longer than a specified threshold. Ultimately we are concerned with determining the number of such attacks and using it as a measure of resilience or lack thereof, in an instance of the shortest-path interdiction problem. We develop and implement algorithms to reduce the required computational effort to solve this counting problem exactly. We illustrate via test cases the impact of different interdiction combinations with regards to the threshold value. Whether these interdictions are random occurrences or intentional, this analysis provides decision makers a tool with which to more completely characterize the resilience of a system of interest.

14. SUBJECT TERMS network interdiction, attacker-defender, defender-attacker-defender, infrastructure resilience, shortest path interdiction	15. NUMBER OF PAGES 69
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2–89) Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

THE THRESHOLD SHORTEST PATH INTERDICTION PROBLEM FOR CRITICAL INFRASTRUCTURE RESILIENCE ANALYSIS

Charles R. Clark
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 2002

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 2017

Approved by: W. Matthew Carlyle
Thesis Advisor

David L. Alderson
Second Reader

Patricia A. Jacobs
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

We formulate and solve the threshold shortest path interdiction problem, which we define as follows: Find a finite set of arcs to attack within a network such that the resulting shortest path from a given source node to a given destination is longer than a specified threshold. Ultimately we are concerned with determining the number of such attacks and using it as a measure of resilience or lack thereof, in an instance of the shortest-path interdiction problem. We develop and implement algorithms to reduce the required computational effort to solve this counting problem exactly. We illustrate via test cases the impact of different interdiction combinations with regards to the threshold value. Whether these interdictions are random occurrences or intentional, this analysis provides decision makers a tool with which to more completely characterize the resilience of a system of interest.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Acronyms and Abbreviations

AD	Attacker-Defender Model
DSPI	Dynamic Shortest-Path Interdiction
FIFO	First-In-First-Out
LO	Lexicographical Ordering
MIP	Mixed Integer Linear Program
PRA	Probabilistic Risk Analysis
SP	Shortest-Path Problem
SPIP	*s*-*t* Shortest-Path Interdiction Problem
SPR	Shortest Path Reordering
TP	Threshold Pruning
ΘSPIP	Threshold Shortest Path Interdiction Problem

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

In this day and age of global inter-connectivity, the function of many interconnected infrastructure systems has become increasingly more important to productivity, transportation, and national defense. Our increasing reliance on these systems exposes us to new risks; the structure of these systems can create situations in which a small number of component losses can have large, systemic impact. To that end, this thesis studies the resilience and vulnerability of an s-to-t shortest path network. We propose models and algorithms providing effective evaluations of the resilience of such a network. The s-t shortest-path problem provides an example that allows us to investigate these models and algorithms on a simple, clean structure that is well understood, however, these models and algorithms are not just limited to this structure and have broader applicability to the entire scope of models for evaluating infrastructure resilience.

We use the definition of resilience provided by Presidential Policy Directive 21: "The term 'resilience' means the ability to prepare for and adapt to changing conditions and withstand and recover rapidly from disruptions. Resilience includes the ability to withstand and recover from deliberate attacks, accidents, or naturally occurring threats or incidents" (White House, 2013). Presidential Policy Directive 21 lays out an immense responsibility for the Federal Government and the many stakeholders involved with our nation's critical infrastructure systems. These goals of reducing vulnerabilities, minimizing consequences, identifying and disrupting threats, and hastening response and recovery efforts related to critical infrastructure require modeling and analysis for which there is no standard. Several tools have been used by many different analysts to provide insights, including probabilistic risk analysis, dynamic programming, and algebraic modeling.

To evaluate this chosen definition of resilience we nominate the Threshold Shortest Path Interdiction Problem (ΘSPIP). Given a threshold, is there an attack of a specified cardinality that causes the s-t shortest path to exceed this threshold? If so, we are then interested in determining how many attacks of a specified cardinality cause the s-t shortest path to exceed this threshold.

In evaluating these questions for attack cardinalities ranging from zero to the number of arcs in the network we derive a resilience chart representing the vulnerability of the network to attacks of each cardinality. With this information, we inform decision makers on the risk of the network to being pushed over the minimum designated threshold.

The primary tool we use for evaluating resilience is enumeration. We are primarily interested in determining the number of ways a network can be damaged to increase its operating cost above a given threshold, and in order to count these cases we enumerate them, and then evaluate the resulting operating cost of the network after damage has occurred.

We devise and present three algorithms that solve this problem and demonstrate a guaranteed reduction in the evaluation requirements of this exponential problem. The Lexicographical Ordering (LO) algorithm, conducting full enumeration is the basis from which we developed the Threshold Pruning (TP) and Shortest Path Reordering (SPR) algorithms. While TP proves to require fewer enumerations than LO, it is not guaranteed to do so. SPR, however will always require fewer enumerations than LO and TP, and in many cases requires significantly fewer, making it the preferred algorithm to use in solving instances of ΘSPIP.

The tools we have developed provide analysts and decision makers information concerning an infrastructure network's resilience with which to determine priorities for investment. With minor modifications, these algorithms and resulting charts can be applied to the vast majority of defender-attacker-defender models of infrastructure resilience, not simply those based on shortest-path models.

Reference

White House (2013) *Presidential Policy Directive 21: Critical infrastructure security and resilience* (Office of the Press Secretary, Washington, DC). https://obamawhitehouse.archives.gov/the-press-office/2013/02/12/presidential-policy-directive-critical-infrastructure-security-and-resil.

Acknowledgments

First and foremost, I would like to thank my wife, Jessica. Her support and love was not only encouraging during my time here at NPS, but also enabled us to have our two beautiful daughters. This tour has brought growth and joy into my life with her by my side. I love you.

I owe a great deal to Dr. Matt Carlyle. Throughout my time here, I have enjoyed his lectures and the many discussions that facilitated going deeper into the material. His mentorship and encouragement while writing my thesis allowed me to accomplish more than I could have thought possible. Without your guidance, support, and insights, this endeavour would not have been nearly as enjoyable and satisfying.

Dr. David Alderson provided a steady hand and encouragement during the thesis process. He enabled me to stay on track and find success. Thank you for all of your help.

Finally, I would like to thank all of the professors and my classmates in the Operations Research Department here at NPS. I have learned so much from all of you both in and out of the class room. I offer my heartfelt thanks to all of you for this wonderful experience.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1: Introduction

In this day and age of global inter-connectivity, the function of many interconnected infrastructure systems has become increasingly more important to productivity, transportation, and national defense. Our increasing reliance on these systems exposes us to new risks; the *structure* of these systems can create situations in which a small number of component losses can have large, systemic impact. A communication network between military units is vulnerable to impacts from non-deliberate (i.e., random) events such as heavy weather, as well as deliberate (i.e., non-random) attacks such as jamming. Similarly, a supply chain could be affected by weather cutting off access for delivery vehicles, or a warehouse fire destroying stock. The ability of these infrastructure systems to withstand or absorb these potential disruptions is referred to as *resilience*. While infrastructure resilience can be defined at different levels, such as the resilience of a single building or piece of equipment, we focus here on systemic impact of loss or damage on the overall function of the entire infrastructure systems. In a road system or communication network, an individual road segment, tunnel, or bridge is a small part of a whole, but the loss of just a few of these can have a significant impact on the entire road network. It is the resilience of this whole that we are interested in.

To that end, this thesis studies the resilience and vulnerability of an s-to-t shortest-path network. How resilient is such a network to random or intentional loss of arcs? To answer this question, we propose models and algorithms providing effective evaluations of the resilience of such a network. The s-t Shortest-Path Problem (SP) provides an example that allows us to investigate these models and algorithms on a simple, clean structure that is well understood; however, these models and algorithms are not just limited to this structure and have broader applicability to the entire scope of models for evaluating infrastructure resilience.

1.1 Infrastructure Resilience

For the purpose of this thesis we use the definition of resilience provided by Presidential Policy Directive 21 (2013): "The term 'resilience' means the ability to prepare for and adapt to changing conditions and withstand and recover rapidly from disruptions. Resilience includes the ability to withstand and recover from deliberate attacks, accidents, or naturally occurring threats or incidents."

Alderson et al. (2014) point out that, since Ancient Rome, major infrastructure systems are paramount to the prosperity and the advancement of society. In that time, the simple road systems and aqueducts were greatly beneficial to the population. Today, we take things like roadways and plumbing for granted, and add to that list power grids, airways, and the telecommunication networks that enable other "infrastructures" such as the Internet, banking, stock and commodity trading, etc. As a society we depend on the function of these systems, and are concerned about their vulnerability to different sources of disruption, both deliberate and random. Events such as Hurricane Katrina in 2005 (CNN, 2017b), the Tohoku earthquake and tsunami in 2011 (CNN, 2017a), airline network outages, and improvised explosive devices in the wars in Iraq and Afghanistan have displayed the damage and interruptions that can be wrought by Mother Nature or a deliberate actor.

While events such as Hurricane Katrina and the Tohoku earthquake may be considered extreme events, thunderstorms knock out power lines and flood streets. Even minor earthquakes can burst sewer pipes and disrupt roads. A simple traffic accident can turn the shortcut to work into a long delay. While these can be considered random occurrences, intentional attacks can cause similar problems. Hackers can potentially disrupt any system controlled by computers attached to the Internet, and hostile actors with explosives can impact physical networks to great effect.

When studying infrastructure systems we choose to use quantitative measures of their *function*. This function is frequently evaluated using a *cost*, which covers all of the operating costs (and penalties, potentially) that are incurred to satisfy any requirements on the system. For example, the cost of operating a road network during rush hour might be the total number of passenger hours expected to get all travellers to their destinations. The cost of operating a fuel distribution system might be the monetary cost of all of the power used to run the pumps, wages for workers, and contract penalties for failing to deliver fuel to a customer. When we evaluate the resilience of such a system, we will be concerned with how this operating cost changes as a result of damage sustained by the system.

The ability to model and analyze these networks and evaluate the impact to operations caused by potential disruptions can inform decision makers concerning response, protection, and investment. This analysis needs to provide more than a single number to evaluate resilience. To that end, we propose to evaluate the resilience of an infrastructure system to a range of attacks, and, more specifically, by asking, "how many attacks of a given cardinality could push the operating cost of our system above a given threshold?" We would like to know how many disruptions at a minimum are needed to push the network over a minimum operating threshold. How many combinations of each cardinality push the network over this threshold?

1.2 The *s*-*t* Shortest Path Interdiction Problem

Most infrastructure systems are complex, and require detailed industry specific models to evaluate their performance. However, many can be modeled fairly accurately using network flow models, sometimes with slight modifications as can be seen in Alderson et al. (2017) and Harris and Ross (1955). We choose to use the shortest-path problem as a surrogate for those models to illustrate our proposed measure(s) of resilience. The basis for this thesis is the *s*-*t* Shortest-Path Interdiction Problem (SPIP) which can be answered using an Attacker-Defender Model (AD). This problem and variations of this model have been explored extensively at the Naval Postgraduate School by the Operations Research Department and work on the subject can be viewed in Alderson et al. (2014, 2015, 2013).

We imagine an *operator* faced with a shortest-path minimization problem and an *attacker* that wishes to maximize the shortest path between the starting and termination nodes. To evaluate, we ask two questions. How many arcs can the attacker "afford" to attack or interdict? Which arcs should be attacked or interdicted to cause the greatest increase to the s-t shortest path? Given the number of arcs to be attacked, we formulate a max-min Mixed Integer Linear Program (MIP) or dynamic programming algorithm determining the exact arcs to interdict or attack, to create the maximized shortest path between the start and terminating nodes.

1.3 Threshold *s*-*t* Shortest Path Interdiction Problem

To evaluate this chosen definition of resilience we nominate the Threshold Shortest Path Interdiction Problem (ΘSPIP). Given a threshold, θ, is there an attack of a specified cardinality, k, that causes the s-t shortest path to exceed this threshold? If so, we are then interested in determining how many attacks of a specified cardinality cause the s-t shortest path to exceed this threshold.

In evaluating these questions for attack cardinalities ranging from zero to the number of arcs (m) in the network we derive a resilience chart representing the vulnerability of the network to attacks of each cardinality. With this information, we inform decision makers on the risk of the network to being pushed over the minimum designated threshold.

1.4 Motivation

A simple setting to motivate this research is the "30 minutes or it's free" guarantee offered in the past by pizza delivery stores. A pizza parlor provides a delivery service to people living within a certain range from the restaurant. The time it takes to make an order once received has a known expectation and variance and therefore, the pizza parlor knows approximately how long it has to deliver the order before the 30 minute time limit is reached.

In determining whether or not to offer this guarantee or how far customers can be from the store to receive delivery, the parlor must consider the expected time to reach all of the customers in range and the variance associated with the drive time. Now we consider disruptions on this network such as road construction, heavy traffic, accidents, etc. and their impact. How many of these disruptions can take place before the pizza parlor will be unable to meet its 30 minute commitment? Are there any customers that can be "cut off" with very few or only one disruption?

Our research provides the tools needed to evaluate this road network and represent its resilience to random disruptions that are outside of the pizza parlor's control. Given this data, the owner of the restaurant can make decisions on offering the guarantee, delivery range of the store, or number of drivers to employ throughout store hours.

CHAPTER 2: Background and Literature Review

Presidential Policy Directive 21 states,

> It is the policy of the United States to strengthen the security and resilience of its critical infrastructure against both physical and cyber threats. The Federal Government shall work with critical infrastructure owners and operators and SLTT [State, Local, Tribal, and Territorial] entities to take proactive steps to manage risk and strengthen the security and resilience of the Nation's critical infrastructure, considering all hazards that could have a debilitating impact on national security, economic stability, public health and safety, or any combination thereof. These efforts shall seek to reduce vulnerabilities, minimize consequences, identify and disrupt threats, and hasten response and recovery efforts related to critical infrastructure. (White House, 2013)

This statement lays out an immense responsibility for the federal government and the many stakeholders involved with our nation's critical infrastructure systems. These goals of reducing vulnerabilities, minimizing consequences, identifying and disrupting threats, and hastening response and recovery efforts related to critical infrastructure require modeling and analysis for which there is no standard. Several tools have been used by many different analysts to provide insights, including probabilistic risk analysis, dynamic programming, and algebraic modeling.

2.1 Probabilistic Risk Analysis

Ezell et al. (2010) write about the use of Probabilistic Risk Analysis (PRA) in assessing terrorism risk. In their writing, they describe threat as the probability of an attack, vulnerability as the likelihood of an attack's success, and consequence as the losses that occur given a successful attack. This leads to an equation of homeland security risk of $Risk = Threat \times Vulnerability \times Consequence$ where threat and vulnerability are probabilities between zero and one.

This is an adaptation of historical uses of PRA, which has been used successfully in random occurrence analyses (Ezell et al., 2010), to cases that might involve deliberate decisions. With regards to infrastructure systems, the analysis follows a simple process. Estimate the probability that a failure, event, or attack occurs; estimate the probability that the failure, event, or attack will be successful if it occurs; and estimate the losses that will occur if the failure, event, or attack is successful; calculate the risk as an expected value.

This process is repeated for all failures, events, or attacks of interest. This information can then be provided to analysts for further evaluation or decision makers for investment or defense decisions. For further details on application to terrorism, see (Ezell et al., 2010).

Brown and Cox Jr (2011) argue that the claim by Ezell et al. (2010), (specifically, that there isn't a fundamental difference between the conditioning on probabilities with an intelligent adversary and random events), is "importantly incorrect." They put forth that "PRA calculations based on this idea can be highly misleading, rather than useful, for terrorism risk." Additionally, they claim that recommendations from this kind of analysis may increase the risk of attacks or fail to reduce them as much as possible. In response, they suggest another approach, "based on explicit recognition that attack probabilities may depend on information that an attacker has but that we do not have" (Brown and Cox Jr, 2011).

In any use of PRA, there is an inherent incompleteness. Every potential event must be modeled, and if not, then it by definition is excluded from consideration. This sort of analysis is limited by the imagination of the analyst(s) and the events considered.

2.2 Dynamic Programming

Sefair and Smith (2016) propose Dynamic Shortest-Path Interdiction (DSPI) as a model for evaluating network interdiction. Along side comparisons to multiple shortest-path interdiction models, they lay out a turn based two player game. In DSPI, instead of an attacker committing all resources prior to the operator selecting a path, they take turns. On each turn, the attacker selects how many resources to use to attack and which arcs to attack. Following this, the operator selects which arc to traverse from his/her current node. This is repeated until the attacker expends all resources or the operator reaches the terminal node.

This model is $\mathcal{NP}$-hard, and becomes computationally expensive for even modest sized problems. Additionally, like other shortest-path interdiction problems, they are limited to finding the best case scenario for the attacker or worst case scenario for the operator for a single level of resources. Given perfect knowledge of the network and the attacker's capabilities, this is useful; however, this does not provide significant insight into a network's resilience as a whole.

2.3 Algebraic Modeling and "Operational Resilience"

Alderson et al. (2015) provide a succinct discussion on the notions of resilience, ultimately concluding that "a key challenge remains how to define resilience in a manner that is (1) quantitative and rigorous enough for objective and precise assessment, (2) flexible enough to capture many facets of resilience already under discussion by researchers, and (3) connected to the operational details of the system under study so that proposed system changes can be naturally evaluated and actually implemented."

In Alderson et al. (2014, 2015), the authors key in on the term "operational resilience" as used in the 2007 National Strategy for Homeland Security (Homeland Security Council, 2007). They "adopt the term explicitly to mean the ability of a system to adapt its behavior to maintain continuity of function (or operations) in the presence of disruptions" (Alderson et al., 2015). They present algebraic models and algorithms built on network flow models including bi-level and tri-level, AD and Defender-Attacker-Defender, and stochastic models. By embedding these models within enumeration algorithms, they are able to establish bounds on the resilience of these systems.

While well proven, this ability to provide a full picture of a network's resilience against all possible attack combinations is limited by the exponential computational requirements, leaving a desire for more efficient algorithms.

2.4 Most-Vital Arcs

Alderson et al. (2013) provide an exploration into the idea of a network's most-vital arcs and, specifically, the idea that there is a simple ranking of arcs from most-to-least damaging to a network. Whether it is a shortest-path or max-flow network, it is easy to believe that there are a set of most-vital arcs. However, this belief is mistaken, and can lead decision makers and analysts to inaccurate assessments of the most important components to protect in a system.

Through the use of simple max-flow network interdiction examples, the authors prove that there is no guarantee for the worst-case arc attacks to be nested as the number of attacks increase. While attacking the arc with the most flow on it when you can only attack one arc seems like a good place to start, that same arc may not be important at all when multiple arcs can be attacked. The closer you get to the min-cut of the network, the potential to move away from large capacity arcs to a distributed set that can cut the network off entirely becomes more likely (Alderson et al., 2013).

This fact complicates the job of decision makers as they consider the impact of more and more damaging attacks. While protecting a single high-capacity arc makes a lot of intuitive sense, if there is a belief that multiple arcs can be attacked or will fail, different choices may be much more appropriate.

CHAPTER 3: Models and Algorithms For Evaluating Resilience

The primary tool we use for evaluating resilience is *enumeration*. We are primarily interested in determining the number of ways a network can be damaged to increase its operating cost above a given threshold, and in order to count these cases we *enumerate* them, and then *evaluate* the resulting operating cost of the network after damage has occurred.

Our "target" model of choice is an s-t shortest path problem because of its mathematical simplicity. We can rapidly evaluate the length of a shortest path, and re-evaluate that length after arcs are damaged, using any of a number of standard algorithms in the literature.

SPs are well-understood and contain structure that can be used to demonstrate techniques with which to reduce the enumeration requirements when addressing our definition of resilience with regards to infrastructure systems. This chapter presents our base models and the corresponding algorithms to solve them.

3.1 Model Basics

The foundation for our models is the classic SP, which asks, what is the cheapest, fastest, or shortest path in a network from a given start point to a given terminal point? More formally, we build upon the notation and definitions from Ahuja et al. (1993). Let $G = (N, A)$ define a directed graph, where N is a set of n nodes, and A is a set of m directed arcs. For each arc $(i, j) \in A$, there is an associated arc length, or *cost*, c_{ij}. In general, we refer to the start node as $s \in N$ and the terminal node as $t \in N$. We define the problem as follows:

Shortest-Path (SP) Problem. Given a network with nodes N, arcs A, arc costs c_{ij}, a start node s, and a terminal node t, find the shortest directed path from node s to node t.

This problem has been studied extensively (see Ahuja et al., 1993, for additional details) and can be solved using a variety of algorithms such as Dijkstra's and FIFO Label Correcting, or as a linear program.

There has been considerable interest in understanding what impact an *interdiction* (i.e., loss, or attack) on an arc has on the shortest-path network, and further, what impact a combination of simultaneous attacks has on the shortest-path network. This problem is known as the Shortest-Path Interdiction Problem (SPIP, see Israeli and Wood, 2002), which is a particular type of AD problem (e.g., Alderson et al., 2014). Following the convention in the latter, we introduce on each arc a penalty, q_{ij}, that is added to its cost if the arc is attacked. That is, an arc is more expensive to use if it is attacked. If this cost is high enough, then the arc effectively becomes unusable for the purposes of the SP problem.

Of interest to us is the cardinality-constrained SPIP, which we define as follows.

Cardinality Constrained Shortest Path Interdiction Problem. Given an SP network, arc penalties, q_{ij}, and a maximum number of arcs to attack k, which k arcs when attacked provide the greatest increase to the operators shortest path and what is the new value of the resulting shortest path?

We present the MIP for this model here, adapted from Carlyle (2016).

Sets and Indices

$i \in N$	nodes (alias j)
$(i, j) \in A$	directed arcs

Data [units]

c_{ij}	cost, or length, of arc $(i, j) \in A$ [cost-units]
q_{ij}	penalty cost to traverse arc $(i, j) \in A$ [cost-units]
k	maximum number of arcs that can be attacked [cardinality]

Variables [units]

X_{ij}	=1 if arc $(i, j) \in A$ on path, =0 otherwise [binary]
Y_{ij}	=1 if $arc(i, j) \in A$ is attacked, =0 otherwise [binary]

Formulation

$$\max_{Y} \min_{X} \quad \sum_{(i,j)\in A} (c_{ij} + q_{ij}Y_{ij})X_{ij} \tag{3.1}$$

$$\text{s.t.} \quad \sum_{j:(i,j)\in A} X_{ij} - \sum_{j:(j,i)\in A} X_{ji} = \begin{cases} 1 & i = s \\ 0 & i \neq s,t \\ -1 & i = t \end{cases} \qquad \forall i \in N \tag{3.2}$$

$$X_{ij} \geq 0 \qquad \forall (i,j) \in A \tag{3.3}$$

$$\sum_{(i,j)\in A} Y_{ij} \leq k \tag{3.4}$$

$$Y_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{3.5}$$

Discussion

The objective function (3.1) includes the decision variables X_{ij} and Y_{ij} where X_{ij} indicates whether or not an arc is on the path and Y_{ij} represents whether or not an arc has been attacked. Minimizing with respect to X_{ij} finds the shortest path, and maximizing with respect to Y_{ij} maximizes the shortest path. Constraint (3.2) ensure balance of flow at each node. Stipulation (3.3) requires that each decision variable, X_{ij}, is non-negative. Constraints (3.4) limit the number of arcs that can be attacked and ensures that the summation of the decision variable, Y_{ij}, does not exceed this limit. Stipulation (3.5) limits attacks to binary representation, preventing "partial" attacks.

In the mathematical formulations, X_{ij} will be 1 if the arc is on the shortest path from s to t, and 0 otherwise due to the minimization of X_{ij} in the objective function. The decision variable Y_{ij}, used in the interdiction model, is a 1 if the arc is attacked, and 0 otherwise. For any fixed attack given by the Y_{ij} values, the resulting shortest path problem is a pure network flow problem, and therefore the X_{ij} values will be integer at an optimal basic feasible solution. Therefore we do not need to restrict those variables to be binary explicitly.

Here, the operator has a simple sub-problem, where he is minimizing the shortest path through the network. The attacker's problem is more complex. There are $\binom{m}{k}$ possible attack combinations. Additionally, the best arc to attack when $k = 1$ may not be part of the best combination of arcs to attack when $k = 2$ (Alderson et al., 2013). In this problem, the number of attack combinations to be considered grows polynomially in m for a *fixed* value of k, but grows exponentially if k grows roughly as $m/2$. If all values of k are of interest, then there are $2^m - 1$ possible attacks to enumerate.

This model solves for the best arc attack combination of cardinality k that the attacker can make to impact the operator. However, we learn nothing about what impact other attack combinations of that cardinality have, nor what impact other cardinalities might have. Further adjustments to the model must be made to evaluate a more robust view of the network's resilience.

3.2 Assessing Resilience

While the SPIP solves for the operator's worst case scenario given a specified k, this does little to provide decision makers with a more complete view of the network's resilience as a whole. Our pizza parlor that is considering a 30-minute-delivery-or-its-free guarantee is concerned with a wide range of interdiction combinations to the road system making up its delivery range. Simply knowing the worst case at a cardinality, or even the worst case at multiple cardinalities does not provide significant insight into the risks associated with the guarantee. A more thorough evaluation of the network is needed to accomplish this.

To provide this insight, we designate a critical threshold, θ, that represents the longest shortest path length the operator will tolerate; any attack that makes the shortest path longer than this threshold is a *critical* attack. An attack that does not result in a shortest path length above the threshold is *non-critical*. Note that the definition of *critical* depends on the value of θ.

Here, we postulate a hypothetical, worst-case attacker who is diametrically opposed to the operator, and therefore desires the resulting shortest path to be greater than this threshold. We nominate the Threshold Shortest Path Interdiction Problem (ΘSPIP) to model this situation:

Threshold Shortest Path Interdiction Problem. Given an SPIP, and an operating threshold θ, is there an attack combination of cardinality k, that increases the shortest path greater than θ?

This problem can then be extended to a counting version that asks, “how many attack combinations of cardinality k increase the shortest path length to a value greater than θ?”

The size of this problem is directly related to the number of arcs in the network, m, and the attack cardinality, k. For a specific k, there are $\binom{m}{k}$ attacks to be evaluated and compared to θ.

3.3 Algorithms

The primary difficulty in evaluating resilience using ΘSPIP is enumeration. Because the number of evaluations to be performed is exponential, we are left to devise algorithms to minimize the evaluations required through *implicit* enumeration, i.e., by accurately accounting for (hopefully) large numbers of attack combinations without explicitly enumerating and evaluating them. We develop three such algorithms for this purpose: Lexicographical Ordering (LO), Threshold Pruning (TP), and Shortest Path Reordering (SPR). The goal is to efficiently reduce the number of evaluations while maintaining *accuracy*, which is defined as correctly accounting for each attack combination with regards to pushing the network over the threshold, θ.

As we present each algorithm, we explain how it executes on the example network presented in Fig. 3.1. This network consists of 5 nodes and 5 arcs. Node s in this example is node 1, and node t is node 5. The threshold, θ, is 2. All arcs have a cost of 1 and a penalty, q_{ij}, of $nC - 1$, where C is the largest arc cost, c_{ij}, in the instance. When attacked, each arc cost becomes 5, making it essentially unusable. When $\theta = 2$, there is only one path that is under the threshold and that is the path containing arcs (1,3) and (3,5).

3.3.1 Lexicographical Ordering

Our baseline enumeration algorithm is Lexicographical Ordering (LO). As its name implies, LO assigns a position to each arc in the network, and then enumerates all attack combinations in a lexicographical order based on the status (unattacked or attacked) of each arc in the combination. Each attack combination is a parent and/or child of another attack combination(s) that it can be related to. This creates a tree of all attack combinations that can be iterated through without risk of re-evaluations. Fig. 3.2 shows a binary representation

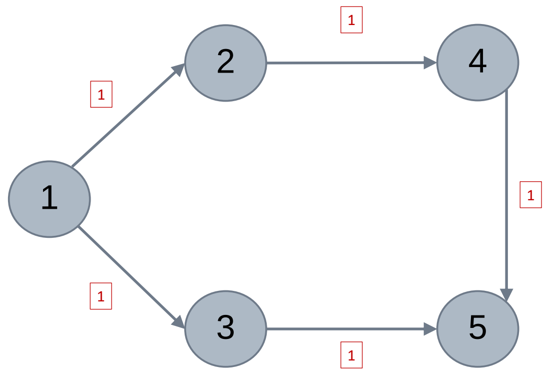

Figure 3.1. Example Network for Algorithm Discussion

example of such an ordering for a five arc network. Fig. 3.3 shows the same example under integer representation.

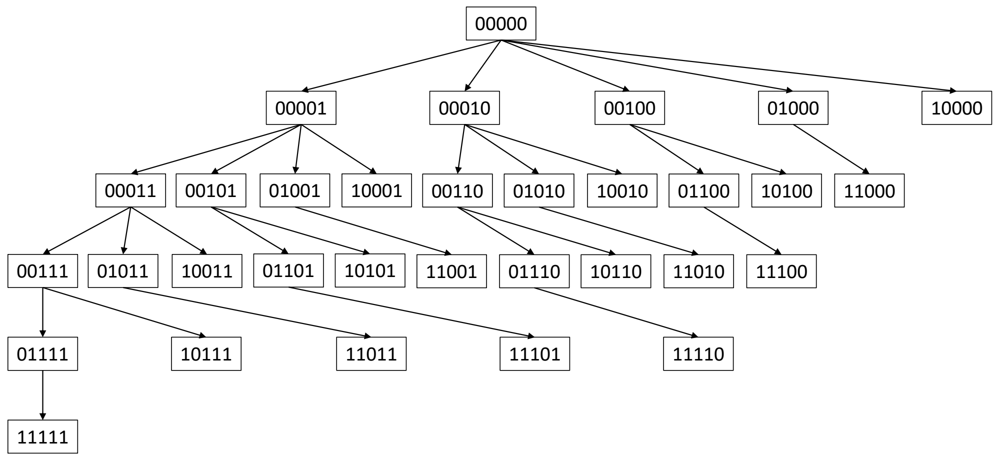

Figure 3.2. Binary Representation of Lexicographical Ordering for a Five Arc Network

To facilitate traversing the tree created by LO, the attack combinations are represented as binary strings and integers interchangeably. The binary representation is viewed such that each zero or one represents an unattacked arc or attacked arc respectively from right to left (ie., 0101 indicates arcs zero and two are attacked). The integer representations provide efficient storage of attack combinations on a queue. A simple conversion of these integers

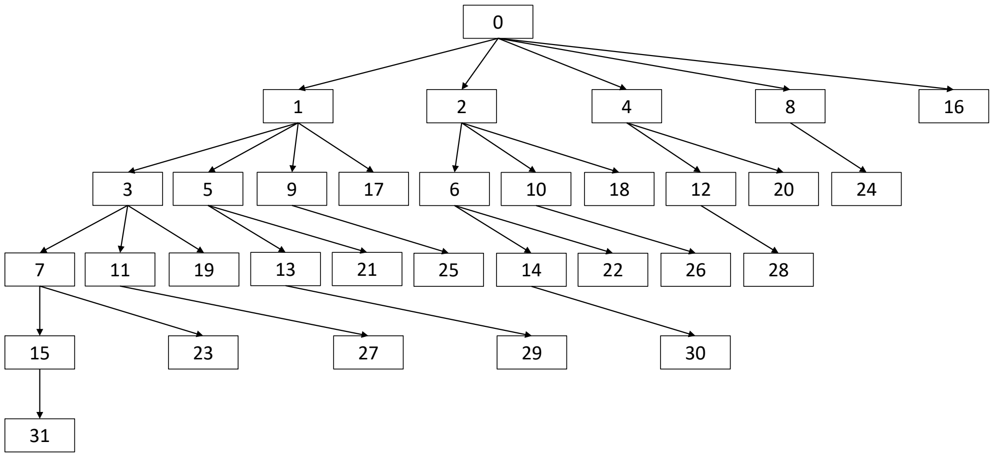

Figure 3.3. Integer Representation of Lexicographical Ordering for a Five Arc Network

to their binary equivalents allows for easy operations.

The tree is "left heavy." The simple structure of this tree is that a node's children are identical to the node with the exception of one of the zeros to the left of the leftmost one in the parent node is changed from a zero to a one. So as seen in Fig. 3.2, the node "00000" has five children, "00001" has four children, "00010" has three children, "00100" has two children, "01000" has one child, and "10000" has zero children. The set of descendants of a particular node consists of all binary representations that agree with the original node in each position after the first '1,' and with every possible pattern filling in the leading zeroes

Initially, the queue is created with the integer zero as the only item in the queue. After an attack combination is removed from the queue and evaluated, its children are created and added to the queue. The algorithm continues this way until all attack combinations have been evaluated in a breadth-first search approach.

LO provides a structure that ensures that all attack combinations are evaluated and no attack combination is evaluated more than once. The number of evaluations required is 2^m. We present Python (v3.5) code used for full enumeration via LO in Fig. 3.4 (Python Software Foundation, 2017). At this point in the code, the required dictionary of arc costs, adjacency list, dictionary of arc penalties, set of nodes, list to track attack combination evaluation results, and assignments of the start and end nodes have been created or accomplished.

We use a first-in-first-out queue, called *queue*, to track and access attack combinations and implement a breadth-first traversal of the enumeration tree. This queue is created using the deque data structure facilitated by the deque module (using "import deque") in Python (v3.5). The queue is initialized by placing attack combination 0 on the queue which represents no arcs attacked. The enumeration then begins via `Loop_1` and continues as attack combinations are placed on the queue and are removed for evaluation until all attack combinations have been evaluated.

Since we are examining many different attacks, and successive attacks in the breadth-first ordering might not be directly related to each other, we create a fresh copy of the attacked arc costs for each evaluation. This is taken care of via `Loop_2`.

In `Loop_3`, we take the integer representation and operate on it, updating the network cost structure based on the attack combination. Using modulus operations (base two) we determine which arcs are to be attacked based on the current value of *combo*. Through these operations, each arc that is attacked gets its cost increased by its penalty cost, or changed to a large number, nC, based on the data provided to the algorithm. Once this is complete, a First-In-First-Out (FIFO) Label Correcting algorithm is used to determine the distance from s to t.

The line of code, `FIFO_Label_Correcting_subroutine()`, represents a subroutine that takes as inputs the adjacency list dictionary, “cost2” dictionary, and start node, “s,” and returns two dictionaries called “pred” and “dist.” The “pred” dictionary contains the predecessor of each node in the shortest-path tree. The “dist” dictionary contains the distance to each node from node s, where “dist[t]” contains the value of the shortest path from s to t.

In `If_Statement_1`, we record whether or not the attack combination pushed the network over the threshold. Finally, `Loop_4` adds all of the children of the attack combination to the queue for evaluation. This process continues while there are attack combinations on the queue.

In reference to our example network from Fig. 3.1, Fig. 3.5 shows the results of evaluation,

```
queue = deque([0])
while queue:                                            #(Loop_1)
    for i in cost:                                      #(Loop_2)
        cost2[i] = cost[i]
    combo = queue.popleft()
    combo2 = combo
    pos = 0
    attacks = 0
    while combo:                                        #(Loop_3)
        if combo%2:
            attacks += 1
            cost2[arcs[pos]] += penalty[arcs[pos]]
        combo = combo//2
        pos += 1

    FIFO_Label_Correcting_subroutine()

    if dist[t] > theta:                                 #(If_Statement_1)
        threshold_attacks[attacks] += 1
    while pos < m:                                      #(Loop_4)
        queue.append(combo2 + 2 ** pos)
        pos += 1
```

Figure 3.4. Full Enumeration Using Lexicographical Ordering

with the red shaded nodes in the minimum spanning tree as those over the threshold and the green shaded nodes as those under or equal to the threshold. As seen, 8 of the evaluations are green, and 24 are red. This means that 24 attack combinations push the network over the threshold and only 8 leave the network operating at or below the threshold. This is done with 32 evaluations. For future reference, the arc labeling is as follows: 0: (1,2), 1: (1,3), 2: (2,4), 3: (3,5), 4: (4,5).

3.3.2 Threshold Pruning

Given the nature of LO, we take advantage of pruning in this tree when an attack combination causes the shortest path through the network to become greater than θ. This pruning reduces the number of evaluations required. The extent to which pruning takes place depends greatly on the structure of the network, primarily the number of attacks required to push the network over the threshold and which arcs need to be attacked to do so at the smallest cardinality.

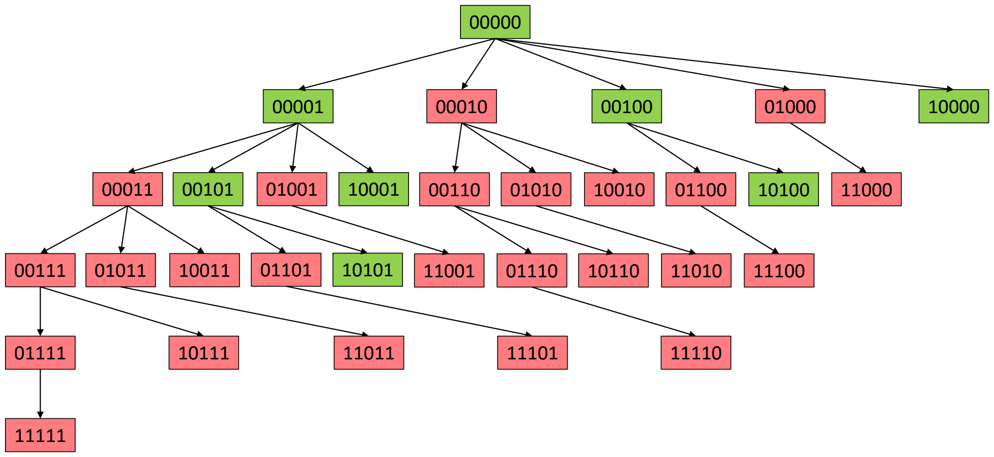

Figure 3.5. Example Evaluation Tree Under Lexicographical Ordering

Since our LO requires that the set of arcs attacked at each descendant of a node contains the attack combination of that node as a subset, the minimum cost of any shortest path created by the descendant attack combinations will be greater than or equal to that of the predecessor node. This *monotonicity* allows us to "prune" the subtree under any node that pushes the network over the threshold. This pruning is accomplished through checking a node's generated shortest path cost and comparing it to θ. Following this check, if the cost exceeds θ, we do not add the children to the queue effectively pruning the portion of the tree below the parent node.

Instead of evaluating all of its descendants, we simply need to know how many of them there are, of each possible attack cardinality, but this is a simple formula based on the number of leading zeroes in the binary representation of the current attack.

To include threshold pruning, we simply replace `If_Statement_1` and `Loop_4` from Fig. 3.4 with `If/Else_Statement_2`, `Loop_5`, and `Loop_6` from Fig. 3.6. In the full enumeration using LO, we simply track the attack combinations that pushed the network over the threshold and their cardinality. With the inclusion of threshold pruning, different actions need to be taken based on whether or not the network has been pushed over the threshold.

`If/Else_Statement_2` and `Loop_5` conduct a simple check if the network is less than

```
queue = deque([0])
while queue:
    for i in cost:
        cost2[i] = cost[i]
    combo = queue.popleft()
    combo2 = combo
    pos = 0
    attacks = 0
    while combo:
        if combo%2:
            attacks += 1
            cost2[arcs[pos]] += penalty[arcs[pos]]
        combo = combo//2
        pos += 1

    FIFO_Label_Correcting_subroutine()

    if dist[t] <= theta:                            #(If/Else_Statement_2)
        while pos < m:                              #(Loop_5)
            queue.append(combo2 + 2 ** pos)
            pos += 1
    else:
        counter = 0
        while counter < m-pos:                      #(Loop_6)
            threshold_attacks[attacks] = threshold_attacks[attacks] + \
                math.factorial(m - pos) / (math.factorial(counter) * \
                math.factorial(m - pos - counter))
            counter += 1
            attacks += 1
```

Figure 3.6. Threshold Pruning Algorithm

or equal to the threshold and if so, adds the children of the attack combination to the queue. This operates equivalently to `If_Statement_1` and `Loop_4` from Fig. 3.4, however it no longer takes place for every attack combination. Due to the parent-child relationship between attack combinations, we do not add the children of those combinations that push the network over the threshold. Instead, we must account for all descendants of the attack combination.

This accounting takes place via `Loop_6`. In this loop, we use a counter for binary position tracking, *m* which remains constant, and *pos* representing the leftmost one in the binary representation of the attack combination that also remains constant. The first pass of the loop will add the newly found attack to the appropriate cardinality tracker. Following loops iteratively add one to the number of attacks and calculates the number of attacks of that cardinality with the parent attack combination as a proper subset. For each cardinality, it is simply the calculation of $\binom{m-pos}{k}$ where k is the iterated cardinality.

Ideally, this pruning will take place early and on the left side of the minimum spanning tree where the greatest reduction of evaluations can take place. However, any amount of pruning is beneficial.

Referencing our example network in Fig. 3.1, we can see the actual evaluation of the network in Fig. 3.7. This figure is identical to Fig. 3.5 with the exception of all children of red nodes having been pruned (unshaded). The 8 green nodes are still present in the tree, however, there are only 6 red nodes shaded. That is due to the fact that the descendants of these nodes have been pruned and tabulated without evaluation. So, the same analysis has been achieved with only 14 evaluations.

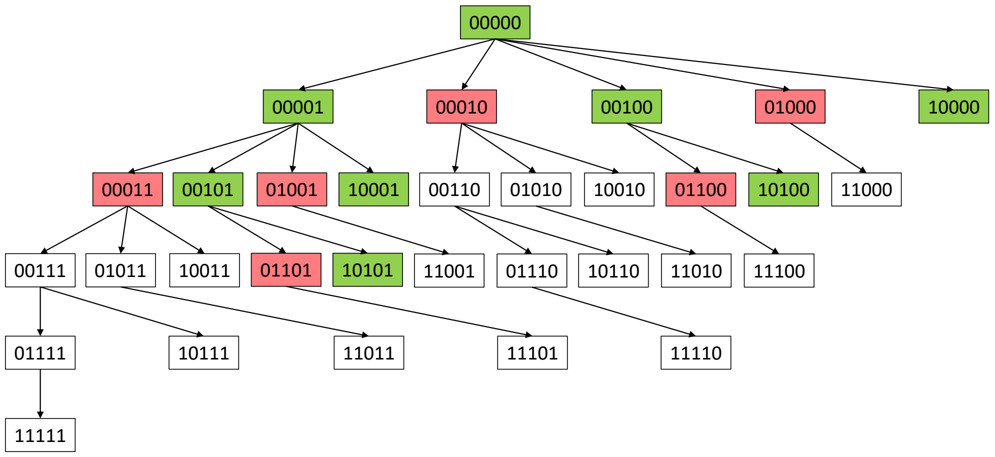

Figure 3.7. Example Evaluation Tree Under Lexicographical Ordering with Pruning.

3.3.3 Shortest Path Reordering

To reduce the amount of enumerations that must be evaluated, we desire to prune as early as possible in the left side of the minimum spanning tree created by LO. Controlling the arc ordering so that the arcs that are likely to be members of the minimum cardinality of attack combinations that cause the SP to exceed θ are on the left side of the tree is a great benefit to pruning. In ordering the arcs, we have designed a heuristic for reordering. This heuristic takes advantage of the fact that in a shortest-path network, to extend the shortest path, an arc on the shortest path must be interdicted. We call this Shortest Path Reordering.

This heuristic is valid at any time during the evaluation process as long as no arcs that have been previously reordered are reordered again. To accomplish this, we must track the ordering of the arcs. We do this with the use of two python lists. The first, referred to as "order", contains the arcs' integer representation in their order. The second list, position, is used to easily access an arc's position. Within this structure, if the current arc of interest is arc 6 which has previously been reordered to the left into position 3, then order(3) returns 6, and position(6) returns 3. This allows for referencing arcs by their LO position, or an LO position by arc number. The final piece to track is a simple integer value representing how many arcs have been renumbered. We use these data structures to swap arcs represented with low integer values, that by labeling are leftward in the LO structure, with arcs represented by high integer values that are on the shortest path.

To accomplish this arc reordering, in Fig. 3.8 we add `Function_1`, `If_Statement_3`, and `Loop_7`. Additionally, there is a minor change to `Loop_5`, generating `Loop_8`. These changes can be seen in Fig. 3.8. `Function_1` is a simple function that takes the distance labels and predecessors generated from the FIFO Label Correcting algorithm and generates the shortest path and returns it as a list of integers representing the arcs on the shortest path following the attack combination.

`If_Statement_3` and `Loop_7` accomplish the reordering. `If_Statement_3` checks to see if all arcs have been reordered, and if so, no more reordering will take place. `Loop_7` takes each arc in the shortest path and checks to see if it has not been reordered. This is accomplished by checking to see if the arcs position is greater than the number of arcs currently reordered. If it is, then its position is swapped with the arc currently in the position of the first arc not reordered.

Finally, the last change is to `Loop_5` from Fig. 3.6. At this point in the algorithm, all arcs on the current shortest path have been reordered. Therefore all arcs in a position greater than or equal to "renum" are not on the shortest path and therefore will not extend the shortest path. Additionally, due to LO, none of their descendants will attack this shortest path either, and so they are all green and do not have to be evaluated. This minor change in the loop definition provides pruning of attack combinations that will never push the network over θ.

The potential benefits from this reordering are that it can be done at every step until reordering is complete and it is simple to implement. Additionally, we are pruning approximately 2^{m-p} arcs where m is the number of arcs in the network and p is the number of arcs on the shortest path at each reordering.

In reference to our example network in Fig. 3.1, after the initial evaluation with no attacks, the shortest path is node 1 to node 3 to node 5. Therefore, we reorder arcs (1,3) and (3,5) to the front of the list, so the arc ordering is as follows: 0: (1,3), 1: (3,5), 2: (2,4), 3: (1,2), 4: (4,5). Since they are the only two arcs on the initial shortest path, all other attack combinations and their children do not need to be evaluated. Additionally, since either attack pushes the network over the threshold, no children of the one arc attacks on (1,3) or (3,5) need to be evaluated. This leaves us with only three evaluations to complete for this network as seen in Fig. 3.9.

This example is a simple one and created for demonstration purposes only. However, it clearly shows the potential benefits of each algorithmic addition. And since each addition does not increase the number of evaluations, only highly robust networks can expect to not see a benefit from their inclusion.

We propose to model the resilience of a system through a parametric analysis of ΘSPIP for k ranging from zero to m using these algorithms. The resulting counts of attacks of size k that exceed the threshold for each value k, reveal how susceptible the network is to increasing numbers of attacked components. However, this analysis is exponential in m, with the exact number of evaluations to be conducted under complete enumeration being 2^m. We present as an example the results of a typical network analysis in what we call a *resilience chart* as illustrated in Fig. 3.10. Once we perform this analysis and summarize the results, measures of interest to us include:

1. The proportion of attacks of size k that exceeds θ
2. The lowest value of k for which at least 1 attack > θ
3. The largest value of k for which at least 1 attack $\leq \theta$.

```
queue = deque([0])
while queue:
    for i in cost:
        cost2[i] = cost[i]
    combo = queue.popleft()
    combo2 = combo
    pos = 0
    attacks = 0
    while combo:
        if combo%2:
            attacks += 1
            cost2[arcs[order[pos]]] += penalty[arcs[order[pos]]]
        combo = combo//2
        pos += 1

    FIFO_Label_Correcting_subroutine()

    path = unpack_path(pred, start, end, inv_arcs)        #(Function_1)
    if renum < m:                                         #(If_Statement_3)
        for i in path:                                    #(Loop_7)
            if position[i] >= renum:
                ii = position[i]
                order[ii], order[renum] = order[renum], order[ii]
                position[order[ii]] = ii
                position[order[renum]] = renum
                renum += 1
    if dist[t] <= theta:
        while pos < renum:                                #(Loop_8)
            queue.append(combo2+2**pos)
            pos = pos + 1
    else:
        counter = 0
        while counter <= m-pos:
            threshold_attacks[attacks] = threshold_attacks[attacks] + \
                math.factorial(m - pos) / (math.factorial(counter) * \
                math.factorial(m - pos - counter))
            counter += 1
            attacks += 1
```

Figure 3.8. Shortest Path Reordering Algorithm

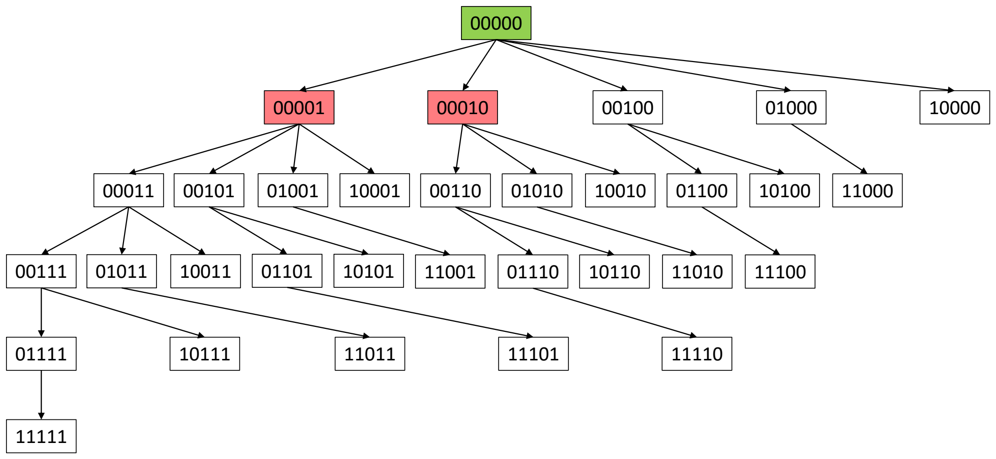

Figure 3.9. Example Evaluation Tree Under Lexicographical Ordering with Pruning and Shortest-Path Reordering.

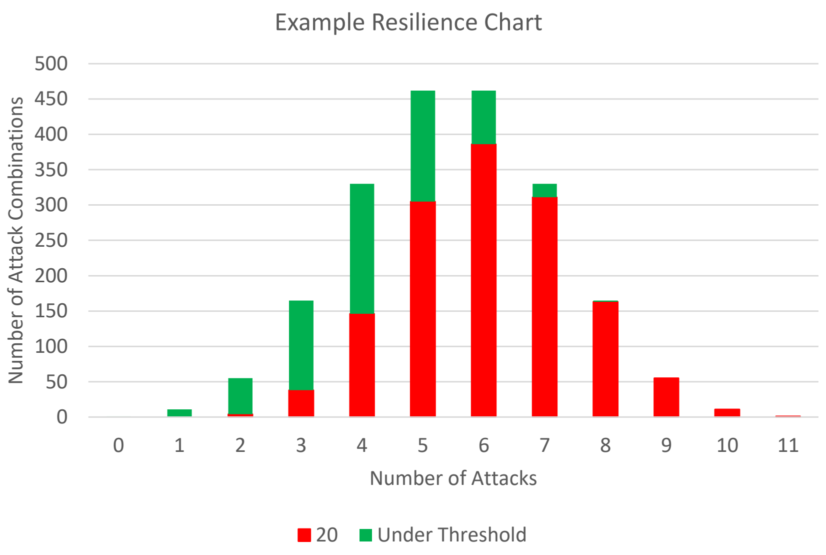

Figure 3.10. Example Resilience Chart For an 11-Arc Network. Red bars indicate the number of attacks of each given cardinality that cause the SP to exceed the threshold, while the green bars indicate the remaining attacks (i.e., those that do not cause the SP to exceed the threshold). For example, slightly more than 300 5-attack combinations exceed the threshold, while the remaining attacks (slightly more than 150) do not.

CHAPTER 4: Results

We provide several small instances of ΘSPIP on which we test our algorithms. We evaluate the performance of each algorithm in terms of the amount of enumeration required for it to solve each instance. We then perform a parametric analysis on the threshold value for each instance to derive insights about the resilience of that network to attack, and to show how to illustrate this resilience through numerical results and informative charts.

We present three figures for each network to summarize the computational results. The first is a directed graph displaying the network structure. The second presents the number of attack combinations enumerated by each of the three algorithms, LO, TP, and SPR, to fully solve that instance of ΘSPIP for each of several threshold values. The third is a histogram called the *resilience chart* of the instance.

4.1 Chapter 3 Example Network

Figure 4.1 is the same as the one used in Chapter 3 as the example and is presented here as an introduction to our analysis. This is a simple graph with 5 nodes and 5 arcs. Node s is node 1, node t is node 5, each arc $(i, j) \in A$ has a cost of 1, and the value of nC for this network is 5. There are only two arc-independent paths. The shortest path includes two arcs and the longest path includes three.

In Fig. 4.2 we see that the TP and SPR algorithms have a distinct advantage in the enumerations required at the lower threshold values. As the threshold increases, the difference in the number of attack combinations enumerated between them and the LO algorithm shrinks, until at the highest threshold, TP performs the same amount of enumeration as LO, but we still see that SPR avoids some enumeration.

This holds true regardless of network data or structure and is due to the pruning of attack combinations that do not include arcs that have been reordered. As mentioned previously, we are able to do this based on the fact that to extend the length of the shortest path through a network, you must attack an arc on the shortest path. These results are typical, but each

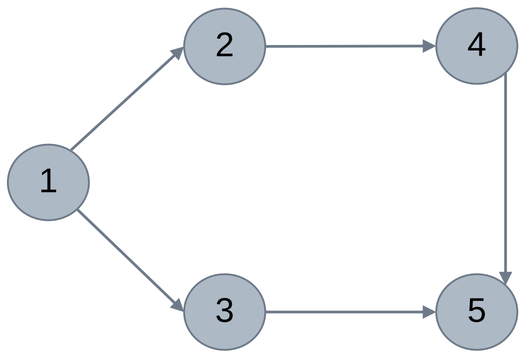

Figure 4.1. Example Network Depiction

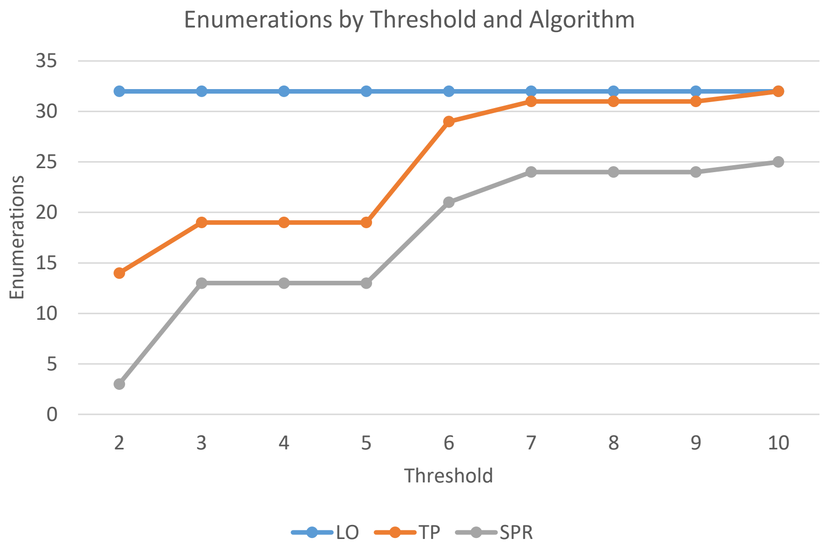

Figure 4.2. Enumerations Required by Threshold and Algorithm (Example Network). While the data for each algorithm is presented as a line, that is for visual purposes only. The dots on each line represent the results for each discrete analysis, and interpolation between dots should not be considered a valid representation.

of the instances we examine will have different specific shapes for these charts.

In Fig. 4.3 we see that as the threshold increases, the number of attack combinations

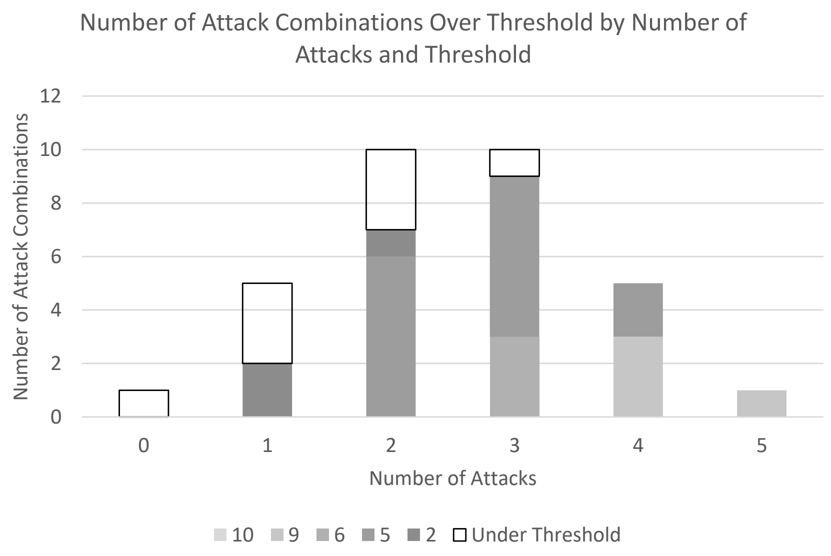

Figure 4.3. Example Network Resilience Chart. The shaded regions in each column represent the number of attack combinations that push the network over the respected threshold at each cardinality of attacks. For a specific threshold all shaded regions representing a threshold of higher value are to be considered included. The un-shaded region of each column represents the number of attack combinations at each cardinality that never push the network over the thresholds of interest.

that exceed that threshold is no larger than before, and is frequently smaller, for each cardinality. Once the threshold reaches a value of 10, there are no attack combinations across any cardinality that push the network over the threshold. Additionally, there are attack combinations of cardinalities zero to three that never push the network over the threshold as long as the threshold is at least the length of the shortest path through the network without any attacks.

Attack combinations that do not contain arcs on the shortest path cannot increase the length of the shortest path, and these attacks are accounted for in the unshaded bars in Fig. 4.3. After the initial reordering these attack combinations are quickly pruned by the SPR algorithm and account for much of the gap seen between the amount of enumeration in TP and in SPR.

In Fig. 4.2 we see monotonicity in the enumerations required as the threshold increases. This directly corresponds to the shaded areas in Fig. 4.3. Where we see an increase in the numerations required in Fig. 4.2 there is a corresponding decrease in the number of attack combinations that push the network over the threshold that can be seen in Fig. 4.3. When there is no change in the enumerations required as the threshold increases, such as thresholds 3, 4, and 5, there is no change in the attack combinations that push the network over the threshold. This is a strictly monotonic increase for both the TP and SPR algorithms, and always holds true.

For TP, we only prune when an attack combination forces the network over the threshold and therefore increasing the threshold can never cause pruning to take place earlier than at a lower threshold. The amount of pruning may remain the same due to the increase in threshold in pruning not leading to an increase in resilience.

For SPR, the reordering is done such that the pruning of attack combinations that do not push the network over the threshold is done based on the shortest path and the fact that you must attack an arc on the shortest path in order to increase the length of the shortest path. For any change in the threshold while the network structure and data remains the same, reordering will always take place in the same way. There may be less pruning of attack combinations that push the network over the increased threshold, but the resultant following shortest path remains the same and therefore the reordering will take place in the same way. Additionally, pruning of attack combinations that push the network over the threshold operates the same as in TP, therefore pruning remains a monotonic function as threshold increases.

4.2 Parallel Network

Figure 4.4 represents a network with 12 nodes and 15 arcs. Node s is node 1, node t is node 12, each arc $(i, j) \in A$ has a cost of 1, and the value of nC for this network is 12. There are five arc-independent paths, none of the paths from s to t share arcs, and all paths have a length of 3. This network is intended to represent purely redundant paths; each s-t path is a perfect substitute for the others, and the more such paths we have, the more resilient we expect the resulting network to be.

In Fig. 4.5 the TP and SPR algorithms each have two values for the range of thresholds investigated. This is due to the network having low sensitivity to the value of the threshold. Since there are multiple redundant paths with equal cost, the attacker must attack each path

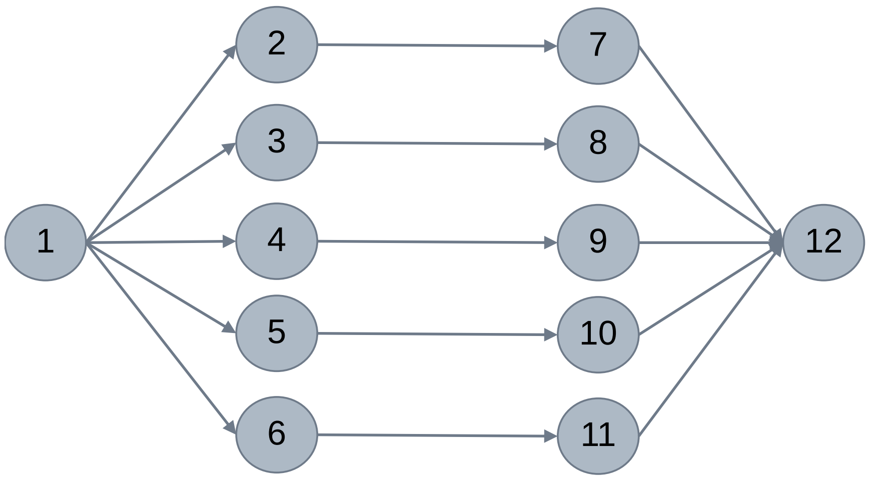

Figure 4.4. Parallel Network Depiction

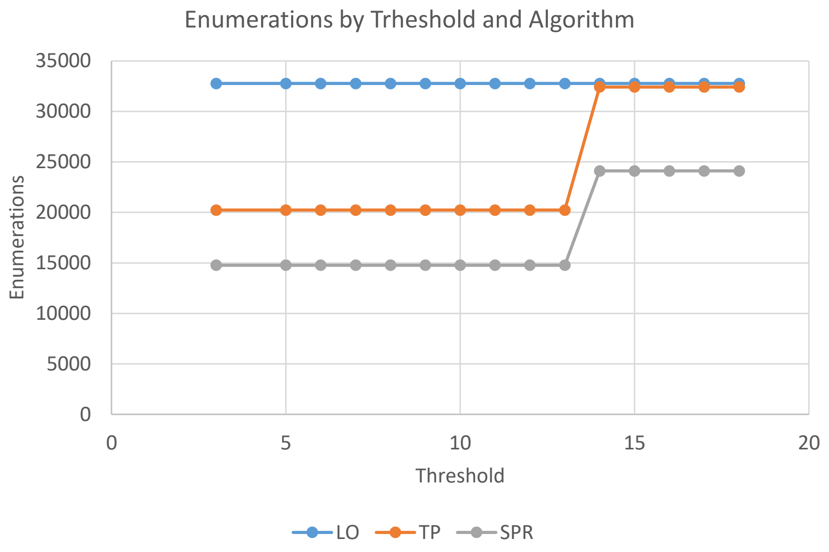

Figure 4.5. Enumerations Required by Threshold and Algorithm (Parallel Network)

before the operator is forced to use a more expensive route. Of note, SPR has a clear advantage at the higher threshold levels due to the fact that the number of arcs on the shortest path is 3 compared to 15 total arcs in the network. SPR is able to prune 2^{15-3} attack combinations due to reordering.

We also see that there is no change in enumeration requirement until the threshold is increased to 14. This is where the operator is able to afford to traverse one attacked arc without exceeding the threshold. This also is visible in Fig. 4.6 where the the second shaded region to be introduced is at 10 attacks, indicating an increased resilience at that threshold. This is where the operator would be potentially forced to traverse 2 attacked arcs.

In Fig. 4.6 the redundancy of the network is clear. There are no attack combinations of cardinality less than 5 that push the network over the threshold and no further implications until the attack cardinality increases to 10. The amount of attack combinations that have no effect on the network is high due to the fact that you have to attack all of the arcs depicted in the graph in a vertical line. This is equivalent to having to apply, at a minimum, a min-cut to the network before you can force the operator over any given threshold

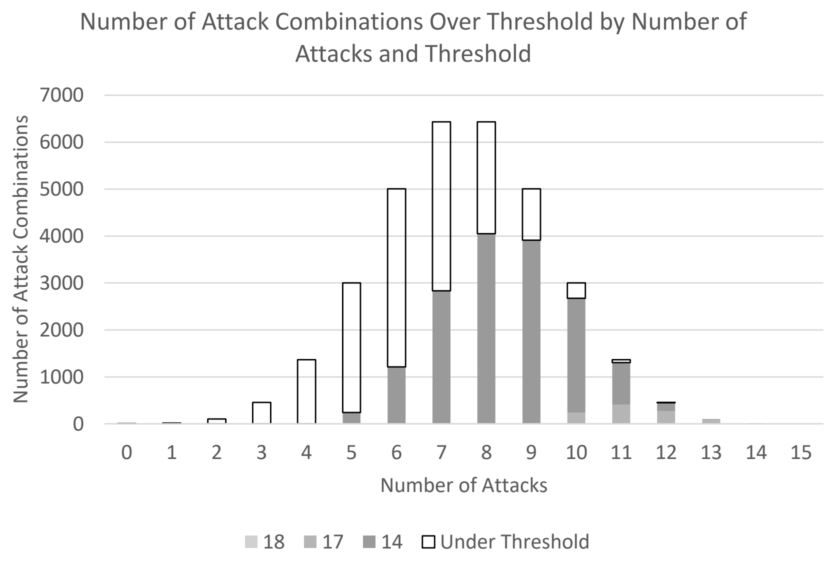

Figure 4.6. Parallel Network Resilience Chart

This network demonstrates a fundamental approach to increasing resilience by building redundancy in the network. If we were to add a sixth arc-independent path to the network, then we would see the first cardinality of attacks that contained any attack combinations pushing the network over the threshold move from five to six. Additionally, we would see an increase in the un-shaded region of each column and a proportional decrease of the shaded region.

4.3 Ladder Network

Figure 4.7 represents a network with two arc-independent paths and the ability to move between paths. It includes 8 nodes and 14 arcs. Node s is node 1, node t is node 8, each arc $(i, j) \in A$ has a cost of 1, and the value of nC for this network is 8. The shortest path includes four arcs and the longest includes 7.

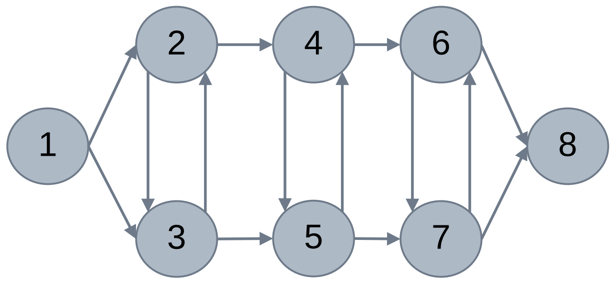

Figure 4.7. Ladder Network Depiction

In Fig. 4.8 there is significant reduction in the number of enumerations required for both TP and SPR with the threshold under 11. Looking at the graph of the network, there are two sets of two arc attacks that guarantee the need to cross an attacked arc, the sets (1,2), (1,3) and (6,8), (7,8). At the minimum threshold of 4, where only the two paths that do not use the interior arcs meet the requirements, any two attacks that attack an arc on the high route and an arc on the low route will push the network over the threshold. This accounts for the early difference in enumeration requirements.

Once the threshold is greater than or equal to 11, the operator is able to afford to cross one attacked arc making pruning much more difficult. Additionally, we can see the enumeration chart repeat its shape. The shape for thresholds 4 to 11 is seen again in thresholds 11 to 18. This indicates two different potential options for increasing resilience. The first is to provide more than two completely distinct paths through the network. The second is to increase the threshold.

In Fig. 4.9 we see that the network is vulnerable at lower thresholds. A significant increase

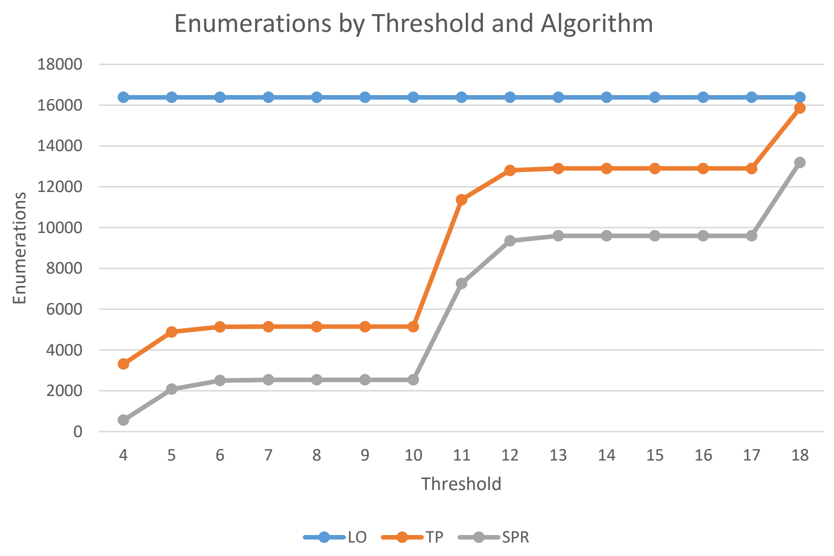

Figure 4.8. Enumerations Required by Threshold and Algorithm (Ladder Network)

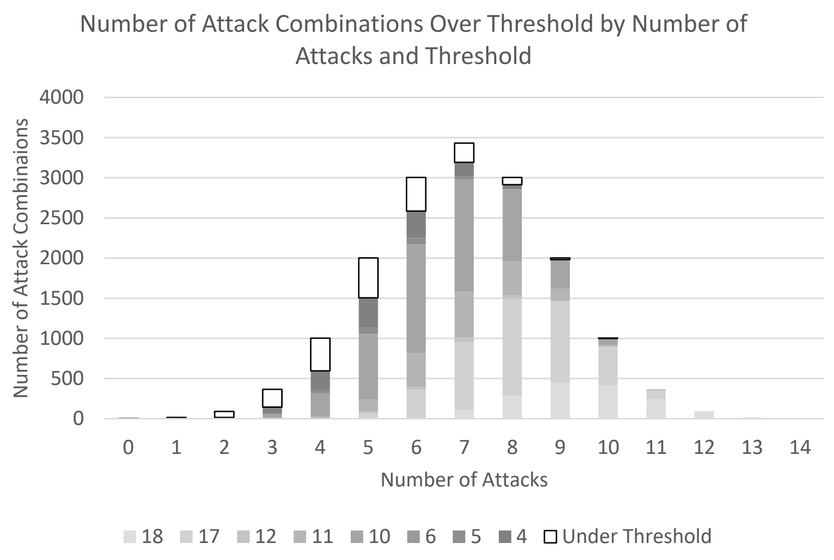

Figure 4.9. Ladder Network Resilience Chart

in resilience takes place once the threshold increases to 11 at which point, the threshold is nearly three times the shortest path cost absent attacks. As discussed previously, this is due to only having two distinct paths through the network leading to low cardinalities of attack combinations having the ability to push the network over the threshold.

4.4 Lattice Network

Figure 4.10 represents a $K_{3,3}$ network with a source and a sink node for a total of 8 nodes and 15 arcs. Node s is node 1, node t is node 8, each arc $(i, j) \in A$ has a cost of 1, and the value of nC for this network is 8. The shortest and longest paths include 3 arcs with 3 arc-independent paths.

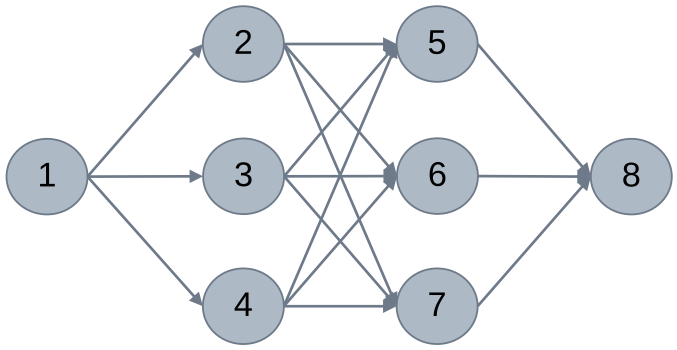

Figure 4.10. Lattice Network Depiction

In Fig. 4.11, while there is a reduction in the number of enumerations required, SPR has much greater success than TP. The pruning that takes place is a result of the limited arcs leaving node s and entering node t. This structure benefits SPR since the paths are short compared to the total number of arcs in the system.

Of interest are the jumps in enumeration requirements at the thresholds of 10 and 17. A threshold of 10 is when the operator can afford to cross one attacked arc, and the threshold of 17 is when the operator can afford two attacked arcs. However, once the threshold is greater than or equal to 10, TP no longer has any significant gains over LO. This is due to the fact that while arcs (1,2), (1,3), and (1,4) might be the first attack that allows pruning and it is in a perfect spot in the tree for pruning, it is the only significant pruning to take place. The dense middle that will lead to little pruning does not add much benefit, and the attack combination of the three arcs leading into node t has no children and therefore no pruning since they are last in the binomial expansion. Meanwhile, with SPR, we get to prune all of the attack combinations that do not include reordered arcs which make up a large portion of the attack combinations as seen in 4.12.

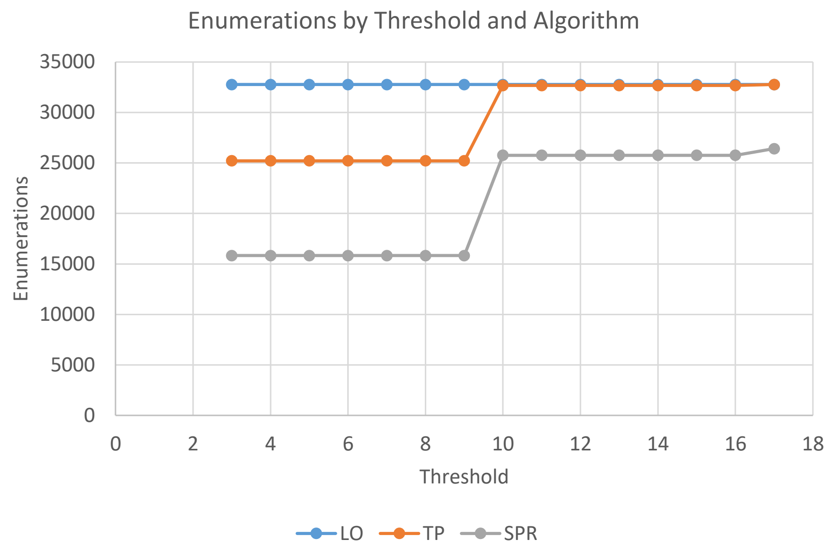

Figure 4.11. Enumerations Required by Threshold and Algorithm (Lattice Network)

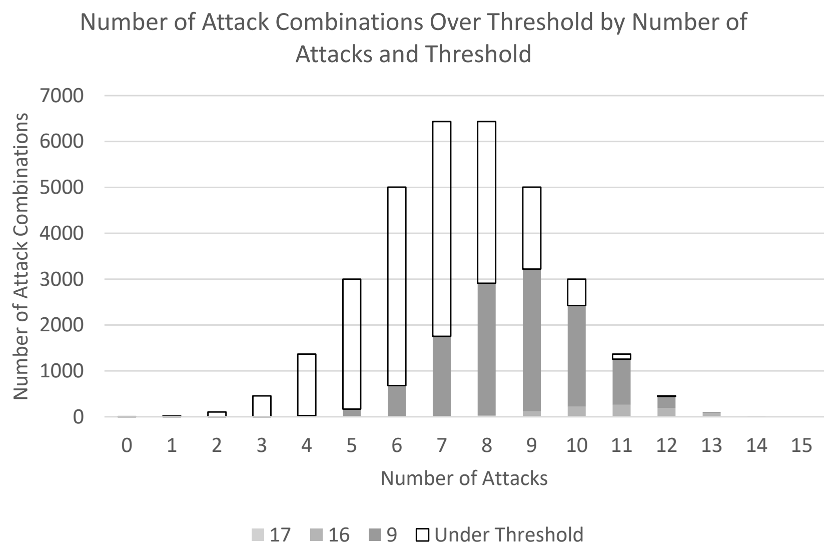

Figure 4.12. Lattice Network Resilience Chart

In Fig. 4.12 the network shows strong resilience. The attacker is required to attack all three arcs leaving node 1 or all three arcs entering node 8 to force the use of an attacked arc. Attacks within the center of the network have little impact due to the number of possibilities for the operator to avoid attacked arcs unless the attacker can attack them all. This emphasizes the impact that choke points can have on a network. While a large portion of a network may be redundant or dense, if there are nodes with low in-degrees or out-degrees that the path must go through, such as node s and node t, they will be vulnerable. While experiencing three random attacks, there are 453 out of 455 that will not push the network over even the lowest threshold depicted. A deliberate attacker may have the knowledge and ability to target those two combinations.

4.5 Lattice 2 Network

Figure 4.13 is an extension of the previous network, adding another layer of dense connectivity that increases the number of nodes to 11 and the number of arcs to 24. Node s is node 1, node t is node 11, each arc $(i, j) \in A$ has a cost of 1, and the value of nC for this network is 11. The shortest and longest paths include four arcs with three arc-independent s-t paths, and there are several ways to create three independent paths of length four.

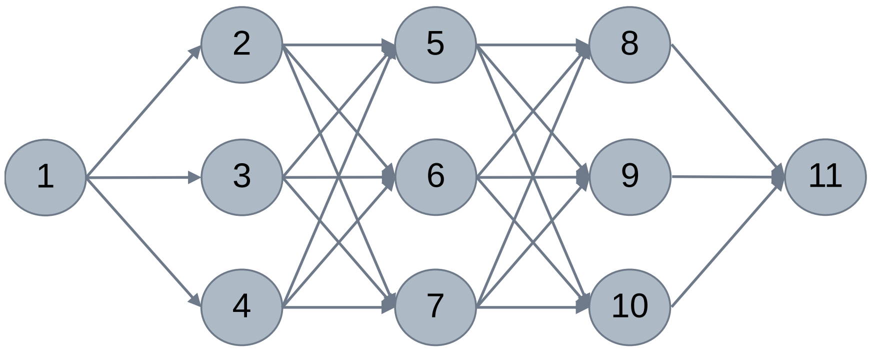

Figure 4.13. Lattice 2 Network Depiction

In Fig. 4.14 we see similar results to Fig. 4.11. The reduction in enumerations by algorithm is less than its predecessor as a proportion of the full enumeration requirements. This would be expected as the addition of the extra layer of 9 arcs increases the number of attack combinations that do not push the network over the threshold and the additional arc on each shortest path reduces pruning from reordering.

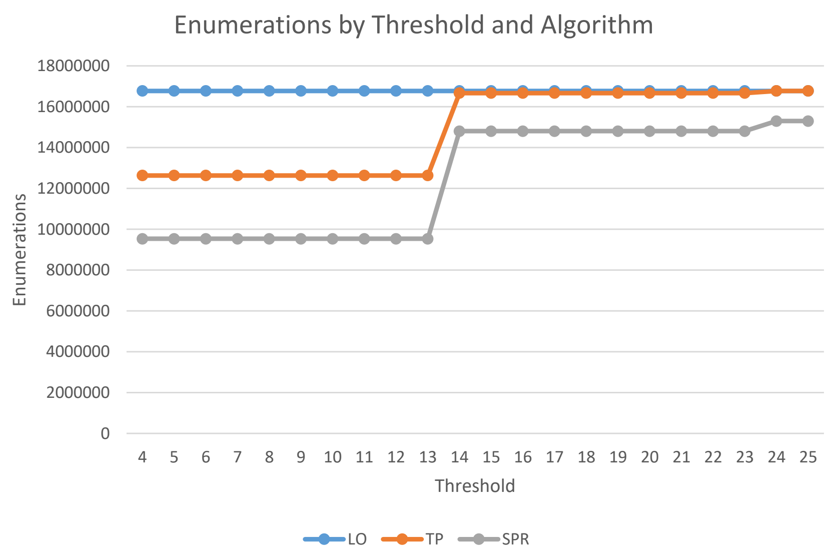

Figure 4.14. Enumerations Required by Threshold and Algorithm (Lattice 2 Network)

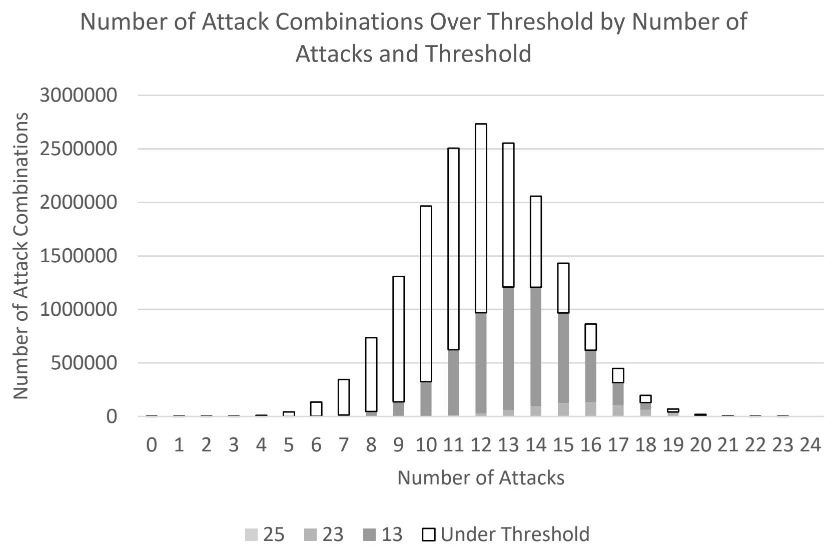

Figure 4.15. Lattice 2 Network Resilience Chart

In Fig. 4.15 the similarities between Lattice and Lattice 2 continue. The resilience charts maintain the same shape driven by the out-degree of node s and the in-degree of node t. The additional layer of arcs here provides additional protection against random interdictions. The driver of the similarities and differences between the two charts is combinatorics. While the out-degree of node s and the in-degree of node t drive the first cardinality of attack that can push the network over the threshold in both networks, the increased layer of arcs in Lattice 2 increases the combinatorial problem size therefore reducing the proportion of attacks that push the network over the threshold at each cardinality.

4.6 Jumper Network

The network in Fig. 4.16 represents a linear network with two arc-independent paths. It includes 7 nodes and 11 arcs. Node s is node 1, node t is node 7, each arc $(i, i+1) \in A$ has a cost of 1 and are vulnerable to attack, each arc $(i, i+2) \in A$ has a cost of 5 and are invulnerable to attack. The value of nC for this network is 100. The shortest path includes 6 arcs with a cost of 6 and the longest includes 3 arcs with a cost of 15.

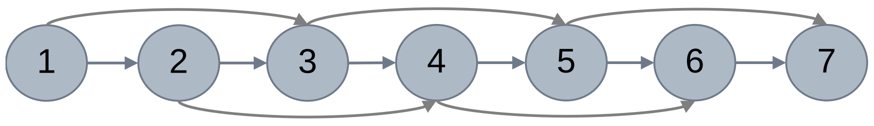

Figure 4.16. Jumper Network Depiction

In Fig. 4.17 the reduction in enumerations required to be evaluated by SPR over TP is relatively low. This is due to the linear nature of the network and the fact that the shortest path when the network is not attacked includes 6 of the 11 arcs. This results in relatively low pruning due to the shortest path. Also of note, when the threshold reaches 15, the operator simply needs to use the 3 invulnerable arcs that create a path from s to t. That can be seen where TP converges to LO at a threshold of 15 and due to the large number of arcs on the initial shortest path SPR is able to prune a relatively small number of attack combinations. All of the increases in enumeration can be accounted for by the threshold's increase to an amount allowing the operator to afford the use of an additional indestructible arc.

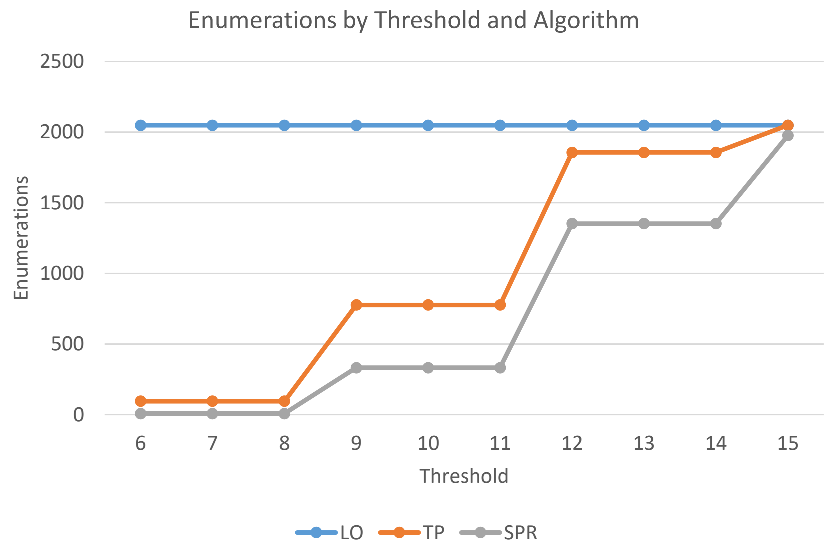

Figure 4.17. Enumerations Required by Threshold and Algorithm (Jumper Network)

In Fig. 4.18 we see that while the network contains an invulnerable path, the cost of that path makes the network heavily reliant on having a high threshold to maintain some level of resilience.

Do to restrictions in the algorithm, the invulnerable arcs are still considered available to attack. However, when attacked, there is no additional penalty cost added. This accounts for the portion of the threshold graph that indicates attack combinations "Under Threshold." Once the number of attacks is increased to 6, there are no longer any combinations that only include the invulnerable arcs.

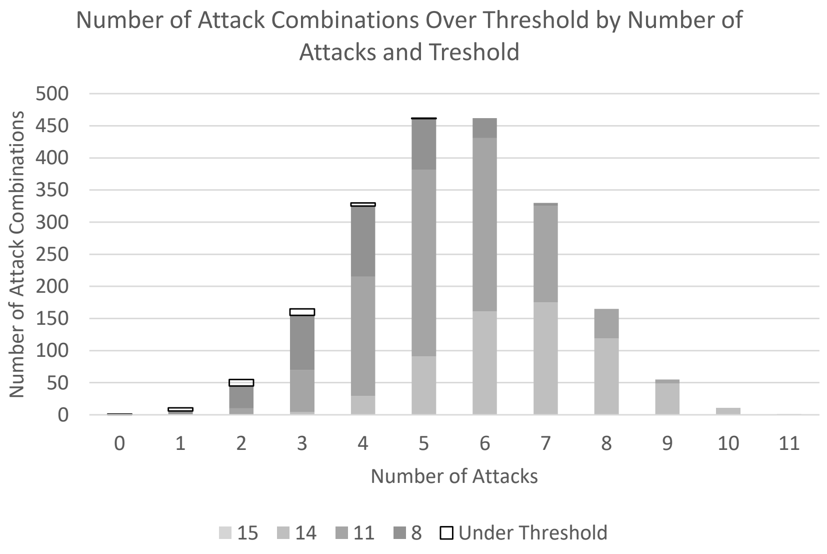

Figure 4.18. Jumper Network Resilience Chart

4.7 Insights

In our case studies and analysis, we see patterns and results concerning network resilience. The parallel network structure in Fig. 4.4 is simple in design, yet shows a relatively high resilience when compared to the other networks in general. The high number of arc-independent paths lead to a requirement for a high number of attacks before the network can be pushed over a threshold. This redundancy is a simple way to increase resilience in a network, however is often times going to be an expensive endeavor.

The existence of highly connected areas in a network has a similar benefit, as seen in Fig. 4.10 and Fig. 4.13. However, this benefit is reduced when considering a well informed adversary. If an attacker has in-depth knowledge of a network structure, the dense portion of the network can be avoided in favor of less dense areas.

Networks such as those in Fig. 4.1, Fig. 4.7, and Fig. 4.16 have neither a high number of arc-independent paths or high density. This leaves the network vulnerable not only to targeted attacks on the choke points node s and node t, but to random attacks throughout the network as well. This can be seen in Fig. 4.3, Fig. 4.9, and Fig. 4.18.

The LO, TP, and SPR algorithms correctly account for all possible attack combinations of a ΘSPIP instance. LO will always enumerate every combination explicitly. TP will only do so in cases in which the "interesting" arcs are the last ones enumerated in the list of arcs, while SPR will only perform a complete enumeration on pathological instances in which the shortest path traverses every arc in the network.

In all instances SPR is guaranteed to require the fewest enumerated attack combinations, and TP will never require more than LO. Even in cases where the threshold is high enough that there are no attacks that push the network over the threshold, the SPR algorithm will always be able to prune those attack combinations that do not include attacks on arcs that have been reordered. This reordering scheme, tied into the LO tree, ensures that SPR will be accurate and always require less enumerations except in the pathological and trivial example of a network where all arcs are on the shortest path. The exact reduction in enumerations will of course vary from instance to instance. The graph structure, threshold, costs, and penalties all affect the order and impact of the attack combinations being enumerated.

SPR performs well on instances with shortest paths that are made up of few arcs in comparison to the total number of arcs in the network. The reordering eliminates a significant number of attack combinations that cannot have any effect on the resulting shortest path length. Both TP and SPR work well on instances with low thresholds and high arc interdiction penalties; this frequently results in low-cardinality attacks that push the shortest path length over the threshold, and lower-cardinality attacks lead to larger pruned subtrees. Networks that include high density sections or multiple redundant paths require more enumeration than sparse networks with few arc-independent *s*-*t* paths. Again, this is because there are usually many low-cardinality attacks that increase the shortest-path length above the threshold.

4.8 A Richer Example

The network in Fig. 4.19 was created to test the different algorithms. The network consists of 6 nodes and 11 arcs and the shortest path touches all nodes. The arcs in the network have varied costs as depicted in the graph and the value of nC for this network is 48. There are only two arc-independent paths. While specific insights are lacking from this example with regards to general network structuring, this example provides a demonstration of the algorithms on a network lacking specific structure and varied costs.

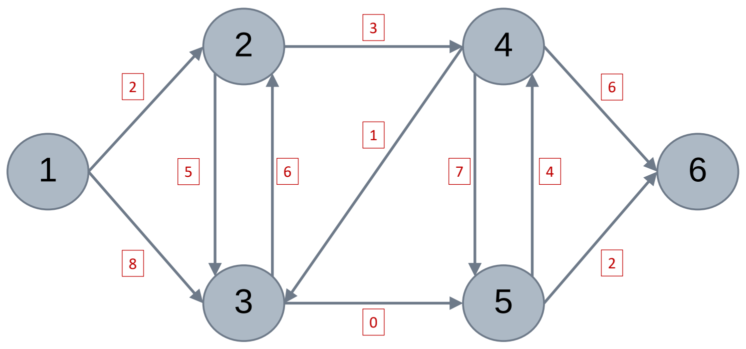

Figure 4.19. Toy Network Depiction

In Fig. 4.20 we get a clear reduction in the number of enumerations evaluated for both TP and SPR. The number of enumerations required demonstrates its monotonicity with regards to threshold and we can see more varied increases with regards to threshold increases.

In Fig. 4.21 the network appears to be fragile. Even at a threshold of 35, which is more than 4 times the cost of the unattacked shortest path we have significant numbers of attack combinations beginning at a cardinality of 4 that push the network over the threshold. With only two completely distinct paths through the network, there are multiple attack combinations that will attack any two distinct paths helping to drive this fragility.

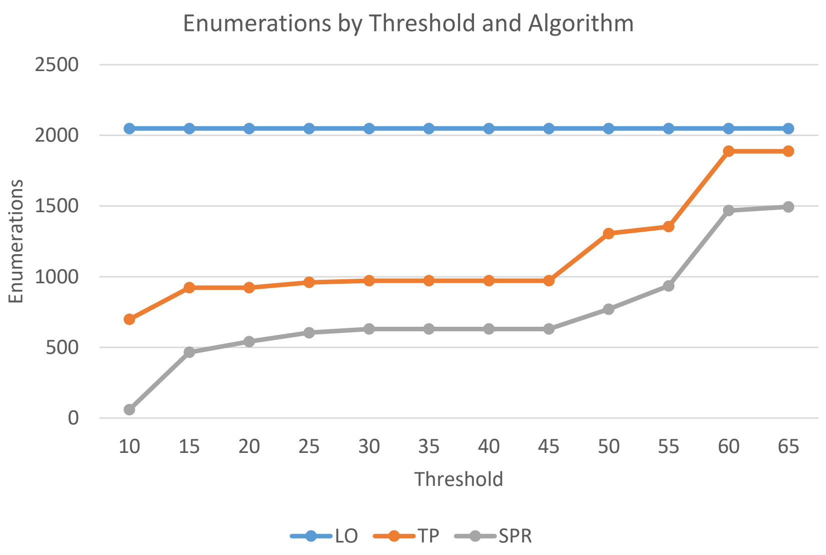

Figure 4.20. Enumerations Required by Threshold and Algorithm (Toy Network)

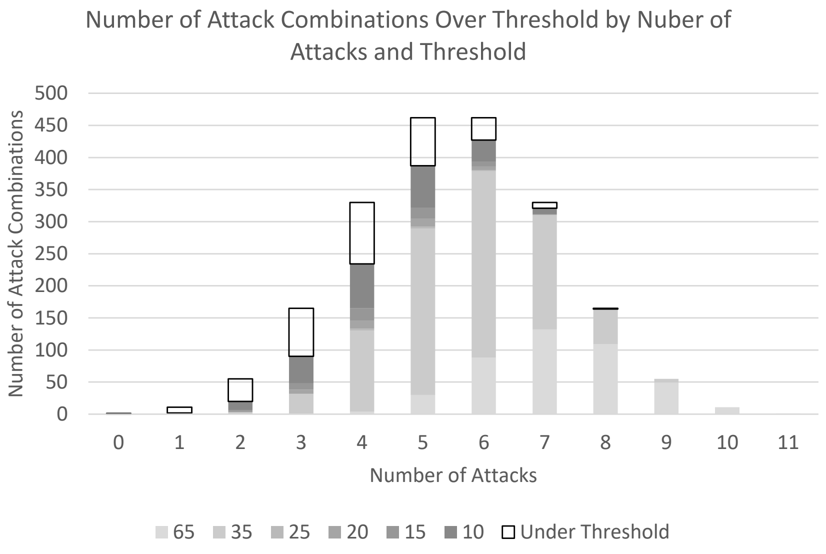

Figure 4.21. Toy Network Resilience Chart

CHAPTER 5: Conclusion

5.1 Summary

We nominate the Threshold Shortest-Path Interdiction Problem, ΘSPIP, and algorithms for solving it as tools for assessing the resilience of a Shortest-Path network to increasing levels of attack. Given a SPIP, an operating threshold θ, and a cardinality $1 \leq k \leq m$, ΘSPIP asks if there is an attack of cardinality k resulting in a shortest path longer than θ. We extend this problem to ask for the number of attack combinations of cardinality k that increase the shortest path length to exceed θ for k ranging from zero to m. This counting problem is exponential in nature, and we have designed our solutions to reduce the required evaluations.

We devise and present three algorithms that solve this problem and demonstrate a guaranteed reduction in the evaluation requirements of this exponential problem. The LO algorithm, conducting full enumeration, is the basis from which we developed the TP and SPR algorithms. While TP has proven to require fewer enumerations than LO, it is not guaranteed to do so. SPR, however, will always require fewer enumerations than LO and TP, and in many cases requires *significantly* fewer, making it the preferred algorithm to use in solving instances of ΘSPIP.

Our results from test cases seen in Chapter 4 demonstrate the reduction in evaluations required by TP and SPR compared to LO on the specific networks. TP and SPR have proven to be monotone, with SPR requiring less than TP in all cases for all thresholds evaluated.

5.2 Recommendations

The analysis we generate through the resilience and enumeration charts, provides a rich picture to decision makers with regards to the resilience of an infrastructure network. As opposed to trying to use a single number to represent the resilience of an entire infrastructure system, our models and algorithms provide a parametric view of resilience, allowing stakeholders to compare the requirements for withstanding attacks or failures of a given cardinality with the cost of defensive investments, or evaluating a particular infrastructure design (or redesign) based on its response to a range of possible attacks or failures.

The use of the SPR algorithm to solve ΘSPIP, and the analysis of its results, enables decision makers to use both quantitative and graphical results to assess the consequences of interdictions and focus limited resources to increase resilience.

5.3 Future Work

There are further extensions to be explored in both the problem formulation and analysis, and in the design and improvement of algorithms to solve those problems.

5.3.1 Parametric Analysis

In Chapter 4, we vary the threshold as a way to demonstrate and evaluate the enumerations required by each algorithm and the number of attack combinations of each cardinality that push the network over the threshold. Similar parametric analyses can be conducted on structure, costs, and penalties.

The tools we have developed can be used to evaluate options with regard to design and various network parameters. A designer is given the job of designing from scratch or redesigning a warehouse supply network to minimize the delivery time of products from a warehouse to customers. The designer might ask, "With the existing or proposed budget constraints, how can I design, or redesign, this supply network to maximize its resilience to random interruptions?"

Using the tools we provide here, direct comparisons can be made between proposed changes in the network. While competing requirements or desires may drive different redesigns, our model and algorithms provide a robust picture of the resilience of the network through the resilience charts following these different redesigns, and allow decision makers to point to specific advantages in a particular design they select.

When requesting or fighting for budgetary funds, our tools can be used to express the difference that can be made with additional resources. If one million dollars provides the means to make changes that result in a certain positive change to the network's resilience, it could be that an increase in funds of 10 percent would drive a different redesign and provide a much greater positive change. While a heftier investment, this foreknowledge can provide decision makers with information needed to justify the expenditure.

5.3.2 Algorithmic improvements

Due to the structure inherent to shortest path networks and our LO tree, there are potential gains to be made through additional algorithmic approaches. While intuition leads us to believe they are worth exploration, they have not been investigated yet.

FIFO Label Correcting Warm Start

Given the impact of potentially enumerating 2^m arc attack combinations, the time required to conduct these evaluations can have a large impact on the time it takes to develop this network resilience evaluation. Any reduction in the time of these evaluations can turn into significant gains for a larger problem, particularly for vast networks. It is in large vast networks where there may be gains to be had from what we call "FIFO Label Correcting Warm Start"

The FIFO Label Correcting algorithm develops a minimum spanning tree view of the shortest path problem for a network. Additionally, it operates off of a queue that methodically moves through the adjacency list of the nodes and updates potential children if there is a shorter path running though this potential parent. Due to this methodology and the parent child relationship of the lexicographical ordering structure, it is possible to calculate the shortest path of the child without conducting a full FIFO Label Correcting evaluation. However, this will require additional memory.

The only change from a parent node to its child node is the addition of one attack. By passing the nodes predecessors and distance labels from parent to child, the FIFO algorithm only needs to update the subtree of node j in the new attack of arc (i, j) if node i is the predecessor of j. To accomplish this, we first must update node j's and its descendants' distance label with the addition of the interdiction penalty associated with arc (i, j). We then look at the reverse star of j and update is predecessor and add all nodes in its reverse star to the FIFO queue. Additionally, for each descendant of j, we must add the member of its reverse star the would provide the shortest path to j if it isn't already in the queue. From here, FIFO can be run and the new shortest path results will be available.

This method can potentially reduce the execution time of a FIFO Label Correcting evaluation. Its ability to do so is determined by the reduced node visitation of the algorithm versus the preprocessing of the FIFO queue. Detailed knowledge of a network's structure and an appropriate test would need to be devised to determine if an attack combination evaluation should do a normal FIFO label correcting algorithm or this Warm Start. Such a test may be a comparison of node j's distance from s and t

Attack Reordering

A potential substitute to shortest path reordering would be to reorder based on attacks that push the network over the threshold. This reordering scheme would wait until an attack was found that would push the network over the threshold and then move the arcs involved in the attack to the right in the binary expansion. This would guarantee the largest amount of threshold pruning possible at the initial reordering. Subsequent reordering isn't guaranteed to provide the largest amount when there are different combinations with the same or close to the same cardinality that push the network over the threshold. Additionally, there is no pruning of combinations that do not push the network over the threshold as in Shortest Path Reordering.

This proposed reordering scheme is expected to perform well in vast networks where the shortest path includes a large number of arcs, yet a small number of attacks are enough to push the network over the threshold. Since the lowest cardinality attack that pushes the network over the threshold is moved to the far left portion of the LO tree and therefore has the largest subtree of all attack combinations of that cardinality, the largest initial threshold pruning is achieved.

The challenge with this reordering scheme is with the lack of a connection between the attacked arcs and the rest of attack combinations within the same breadth of the LO tree. SPR is based on the fact that you must attack an arc on the shortest path to extend its length and therefore attacks not including any of those arcs will never have an effect. This allows SPR to take place at any time, even mid-breadth. Attack reordering would not have the same benefit. The reordering could feasibly result in combinations being evaluated multiple times, double counting, or combinations never being evaluated. An algorithm executing this reordering scheme must account for this to succeed.

Parallel Processing

Parallel processing of this algorithm is possible due to the ability to partition the tree. Since we create a minimum spanning tree of all attack combinations via lexicographical ordering, we can partition the subtrees as desired. For instance, each subtree beginning with a one arc attack could be processed by different processors.

The greatest challenge in parallel processing is to do it efficiently. When viewing the LO tree, the subtrees of each node in a breadth are of different sizes getting smaller from left to right. To simply assign a subtree to a processor will leave many processors complete well before others. Additionally, our algorithm resets the network cost structure every evaluation. This would require every processor to maintain two copies of the cost dictionary or communicate with a master processor to reset the costs each evaluation. Any parallel implementation has to address these utilization and memory concerns.

5.4 Final Comments

The tools we have developed provide analysts and decision makers information concerning an infrastructure network's resilience with which to determine priorities for investment. With minor modifications, these algorithms and resulting charts can be applied to the vast majority of defender-attacker-defender models of infrastructure resilience, not simply those based on shortest-path models. We used the shortest-path structure as a clear example with which to demonstrate their benefit. There is nothing to prevent these algorithms from being used to evaluate the resilience of electric power systems, water distribution networks, fuel supply systems, military logistics, and the continually-expanding realm of cyber systems.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

Ahuja RK, Magnanti LB, Orlin, JB (1993) *Network Flows: Theory, Algorithms, and Applications* (Pearson, Upper Saddle River, NJ).

Alderson DL, Brown GG, Carlyle WM (2014) Assessing and improving operational resilience of critical infrastructures and other systems. *Bridging Data and Decisions* (INFORMS, Catonsville, MD), 180–215.

Alderson DL, Brown GG, Carlyle WM. (2015) Operational models of infrastructure resilience. *Risk Analysis* 35(4) 562–586.

Alderson DL, Brown GG, Carlyle WM, Cox Jr LA (2013) *Sometimes There is No Most-Vital Arc: Assessing and Improving the Operational Resilience of Systems* (Naval Postgraduate School, Monterey, CA*).*

Alderson DL, Brown GG, Carlyle WM, Wood RK (2017) Assessing and improving the operational resilience of a large highway infrastructure system to worst-case losses. *Transportation Science.*

Brown GG, Cox Jr LA (2011) How probabilistic risk assessment can mislead terrorism risk analysts. *Risk Analysis* 31(2) 196–204.

Carlyle, WM (2016) Shortest path network interdiction. Lecture. Network Flows and Graphs, April 27, Department of Operations Research, Naval Postgraduate School, Monterey, CA.

CNN (2017a). 2011 Japan Earthquake - Tsunami Fast Facts. Accessed September 21, http://www.cnn.com/2013/07/17/world/asia/japan-earthquake---tsunami-fast-facts/index.html.

CNN (2017b). Hurricane Katrina Statistics Fast Facts. Accessed September 21, http://www.cnn.com/2013/08/23/us/hurricane-katrina-statistics-fast-facts/index.html.

Ezell BC, Bennett SP, VonWinterfeldt D, Sokolowski J, Collins AJ (2010) Probabilistic risk analysis and terrorism risk. *Risk Analysis* 30(4) 575–589.

Harris TE, Ross FS (1955) *Fundamentals of a Method for Evaluating Rail Net Capacities*. RAND Corp, Santa Monica, CA.

Homeland Security Council (2007) *National strategy for homeland security* (Department of Homeland Security, Washington, DC). https://www.dhs.gov/national-strategy-homeland-security-october-2007.

Israeli E, Wood RK (2002) Shortest-path network interdiction. *Networks* 40(2) 97–111.

Python Software Foundation (2017) *Python v3.5*. Software. Python Software Foundation, Beaverton, OR. https://docs.python.org/3.5/reference/index.html. Accessed September 13, 2017.

Sefair JA, Smith JC (2016) Dynamic shortest-path interdiction. *Networks* 68(4) 315–330.

White House (2013) *Presidential Policy Directive 21: Critical infrastructure security and resilience* (Office of the Press Secretary, Washington, DC). https://obamawhitehouse.archives.gov/the-press-office/2013/02/12/presidential-policy-directive-critical-infrastructure-security-and-resil.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California